Journal of Computer Science

Original Research Paper

Design and Development of an Automatic Penetration Test Generation Methodology for Security of Web Applications

Shilpa R. G., Pushphavathi T. P. and Murthy P. V. R.

Faculty of Engineering and Technology, M. S. Ramaiah University of Applied Sciences, Bangalore, India

Article history
Received: 16-03-2024
Revised: 17-05-2024
Accepted: 29-05-2024

Corresponding Author:
Shilpa R. G.
Faculty of Engineering and Technology, M. S. Ramaiah University of Applied Sciences, Bangalore, India
Email: shilparg.ms.mc@msruas.ac.in

Abstract: In today's world, web application security is becoming more crucial. Web applications frequently hold sensitive data, which might be compromised if it were to fall into the hands of a hostile attacker. This leads to significant losses for businesses and customers alike and exposes the qualities of confidentiality, integrity, and availability. A penetration test is an attempt to exploit vulnerabilities in an IT infrastructure with the goal of evaluating its security. Existing methodologies do not have a systematic basis to represent information gathered hence creating automatic attack generation difficult. The proposed model-based penetration test framework provides a repeatable, systematic approach for conducting penetration tests based on appropriate models of the behavior of the web application. It incorporates a novel approach for model-built security tests along the two scopes of vulnerability coverage criteria and automated attack generation from vulnerability mapping and abstract behavior of web applications. The algorithms are proposed for both manual and automatically driven penetration tests from the state models. The proposed approach is illustrated on a web app location within the banking sector exploiting input validation vulnerabilities.

Keywords: Penetration Testing, Vulnerabilities, SQL Injection, Secondary SQL Injection, Client-Side Manipulation, Model Driven Testing, State Models

Introduction

The growing dependence on online applications for a range of functions, from basic webpages to intricate web applications managing confidential information, has made the security of web applications a critical problem. In today's world, web application security is becoming more and more crucial. Web applications are susceptible to a surplus of security vulnerabilities, which malicious hackers can exploit to compromise their availability, integrity, or confidentiality. Insufficient input validation outlooks out as a significant security apprehension for web applications Li and Xue (2014). According to the study carried out by Cisco dated March 2, 2023, India ranks second in terms of all breaches disclosed in 2022. Ransomware was the cause of 33% of the attackers, while unprotected databases in India accounted for 17% of cyber-attacks. According to the study carried out by positive technologies the applications managing government data accounts and are the targets of the most attacks. The percentage of attacks that targeted web applications rose from 14-23 percent in comparison to 2020. This is most likely a result of the rising quantity of data in government information systems and the expanding number of online services available. The number of breaches increased by 83% of data breaches in 2022, 59% of data breaches have been exposed on social security and mobile users have become victims of mobile cybercrime within the past year. Along with government institutions, the most vulnerable websites are the bank's web applications. Critical vulnerabilities were found in over 89% of the financial institution's systems.

To safeguard the security and consistency of web-based systems, developers and administrators must systematically address these vulnerabilities through rigorous security methods and constant monitoring Awang and Manaf (2013). Vulnerabilities fall into three categories: Input validation vulnerabilities, session management vulnerabilities, and application management vulnerabilities (Choiriyah and Qomariasih, 2023).

Developers and security experts need to be conscious of these weaknesses and depend on robust security practices, including secure coding techniques, periodic security assessments, and the execution of appropriate security approaches, to be able to manage these risks effectively.

By cautiously endeavoring to exploit weaknesses, a penetration test is an attempt to assess an IT system's security. Bacudio *et al*. (2011) discuss the benefits,

strategies, and techniques of carrying out penetration tests. A penetration testing methodology which is divided into four stages namely information gathering, attack generation, exploitation, and reporting the vulnerabilities is suggested by the author. The paper lacks the approach for adequacy criteria for penetration tests. The continual occurrence of vulnerabilities has resulted in an increase in demand for approaches that may identify vulnerabilities in deployed web-based applications. As a consequence, in order to be able to prevent/detect potential vulnerabilities and security issues, it is essential to enhance testing techniques as well as the effectiveness of significant processes. Penetration testing is crucial for effectively discovering and fixing vulnerabilities in web applications, which helps organizations strengthen their safety measures and decrease the probability of cyber-attacks Benikhlef *et al*. (2016).

Currently, a significant gap in penetration testing is in the area of process steps and methodologies for attack generation so as to form a sound basis for penetration test design. A related gap is that no models are used to represent information gathered thereby making automatic attack generation difficult.

Model-based design is incorporated in Model-Based Testing (MBT) in order to systematize test activities or model test artifacts Schmidt *et al*. (2016). At the abstract model level, MBT enables preliminary generation and automatic validation of tests. An enormous number of model-based testing techniques used today incorporate into account the automated creation of test cases from a functional description of the system. Existing penetration methodologies are insufficient to generate automatic penetration tests from the information gathered. Model-based testing lowers the amount of expertise needed for security testing and improves the level of abstraction in various aspects, it provides a systematic approach to gather the information for automatic test generation. MBT recycles functional system information, allowing the test engineer to abstract from numerous factors in this regard. But in order to validate security requirements, the testing professional also needs to be fairly knowledgeable in security in order to generate tests Felderer *et al*. (2011) as penetration/security testing is tightly coupled with attack, attack models can possibly fill this gap. Based on the research gaps following are the research questions to be addressed in order to fill the gap.

Research Questions

1. How do we arrive at the mapping of existing tools to known vulnerabilities? How to integrate this mapping into the relevant penetration test process?
2. How can penetration tests (penetration-related sequence diagrams) be automatically generated?
3. What is an effective way of forming a knowledge base or model representation from the information gathered?
4. How can attacks be generated automatically from a model of information gathered?

This study proposes a repeatable, methodical model-based penetration test basis for conducting penetration tests built on appropriate models of behavior of the web application. It is predominantly a manual process step from the information-gathering phase to that of attack generation. There exists a strong need for effective automatic penetration test generation while a lot of work is done in penetration test automation. Systematic methods of representing a knowledge base of vulnerabilities or models for them in the context of penetration test design are also not reported. Current trends are more about penetration test automation (execution of attacks/scripts) but not about automatic penetration test generation. By addressing these research concerns, penetration testing could make major advances that would make vulnerability detection and attack scenario creation more effective and efficient. Furthermore, by incorporating automation and model-based approaches into the penetration testing procedure, it may be possible to address current knowledge gaps and improve testing procedures.

Related Work

For the past era, Model-Based Testing (MBT) has remained a hot topic of significant research interest and some recent studies have discovered some real benefits that come from using it in daily life. Web application security is highly endorsed and safeguarded by penetration testing. Vulnerabilities are identified by testers in a web application by simulating attacks. In direction to achieve this effectively, testers depend on automated methods, which collect input vector data around the targeted web-based application. The efficacy of the attack is determined by the application's responses. The current techniques for accomplishing these processes are frequently inadequate exposing untested and vulnerable portions of the web-based application unidentified.

Model-based Security Testing (MBST), which comes under the broad area of MBT, concentrates more specifically on the System Under Test (SUT's) security requirements, particularly those associated with validation, authorization, privacy, and reliability of data Lunkeit and Schieferdecker (2018). MBST approaches may address concerns concerning the security testing methods used nowadays. Accordingly, early testing through development and process automation during security testing has been rendered possible by MBST Felderer *et al*. (2016). The paper also indicates possible directions for future research, like testing in combination with security and safety or prioritizing attacks based on known vulnerabilities. Sommer *et al*. (2023) discuss

model-driven approach to security testing. The automobile attack database comprising 361 attacks is used to analyze the model for likely attack paths constructed on real-world attacks. Sommer *et al*. (2021). As a result, attacker privileges can precisely represent crucial attack vectors within automotive networks as illustrated in the work. Casola *et al*. (2024) propose the approach, that developers may automatically generate a security test procedure while receiving a set of appropriate security tests that they can replicate for their apps which saves time and effort typically needed for penetration testing tasks, asset identification, and mitigation of test-failures. However, the suggested method relies on catalogs and a security data framework that may be used to formalize the knowledge of security experts.

Lonetti *et al*. (2023) propose IoT system validation spanning a broad spectrum of test ideas in developing application domains that can be enhanced by MBST. MBST devices in core IoT domains have shown to be efficient at evaluating the system's security against an array of popular IoT attacks. The research also promises to combine model-based security testing through further security test techniques, namely fuzz or penetration testing. Halfond *et al*. (2011) suggest a novel approach that addresses the disadvantages of the penetration testing methods currently in existence. This approach identifies an attack that can successfully target a web application and develops input vector identification using two newly developed sophisticated evaluation techniques. Attacks are simulated automatically by input vector data but this approach lacks automatic generation of penetration tests from the input vector data.

A model-driven repeatable, meticulous, and affordable technique of Web application penetration test framework unified into a Security-Oriented Software Development Life Cycle is proposed by Xiong and Peyton (2010). The Recommended framework is an informal methodology and does not ensure a systematic approach for penetration test generation. Also, it fails to define the vulnerability coverage that the framework can achieve. Model-based testing methods commonly address functional features. The version of this method to vulnerability testing that proposes refining the accurateness and correctness of testing is proposed. Lebeau *et al*. (2013) propose a behavioral model and test patterns as the foundation for a Model-Based approach to testing vulnerability that targets to address both logical and scientific vulnerabilities. This approach is not widely used for security testing. A model-based penetration test framework that provides an entirely integrated approach within the system development life cycle suggests a dependable, systematic, and economical approach for web applications. Web penetration testing models are assessed using TTCN-3, the test specification language. Further, Stepien *et al*. (2012) revealed merging distinctive simulations for the appropriate web vulnerabilities and application functionalities caused by a web abstraction model, and a TTCN-3 test framework model is demonstrated. The model fails to address Vulnerability test coverage which can be the basis for prioritizing the penetration tests. MBST is based on formal methodologies, but first, the security specialist is required to develop a suitable web application model. Penetration testing can be instead successful, but the security analyst's experience is crucial. To bridge the gap between these two security testing methodologies, Peroli *et al*. (2018) present MobSTer, an MBST framework that is formal and flexible. The basic idea is that model-checking methods allow an analyst to execute security testing disregarding crucial tests by automating the process of accessing potentially vulnerable areas in the web application. The authors highlighted that employing model-based security testing has several advantages. But for real advantage an exhaustive approach to security modeling and testing is essential.

Peleska *et al*. (2018) presented practical benefits, such as automated requirement traceability, continuous test method regeneration in regression testing and more logical and effective test result analysis. Garousi *et al*. (2021) provide intangible but important benefits, MBT improved test case design in a measured investigation with a software testing corporation compared to the previously used model-free test scripts which resulted in increasing the accuracy of fault detection. MBST presents the same benefits as MBT. As illustrated by Peroli *et al*. (2018) pertaining to security testing, the prototypes are essential to be augmented with the security objectives that the SUT must adhere to. However, the primary advantage of these security-enriched models is their reusability Murthy and Shilpa (2018).

Design and Development

Existing Penetration Testing Methodology

A particular methodology to identify vulnerabilities in web-based applications is penetration testing. Through assaulting the applications employing automated tools or manual techniques, it attempts to take advantage of vulnerabilities in a web application by the attacker or an unauthorized user. Typically, penetration reports provide a summary of a list of vulnerabilities that were found. However, this technique is incomplete and falls short since there are normally no morals that stipulate which penetration tests to execute and what inputs to try.

Information gathering, attack generation, and response analysis are the three stages of the current penetration testing methodology. An overview of all three phases is shown in Fig. 1. Penetration testers use a variety of tools and tactics to gain information about a certain web application at the time of the information-gathering phase. Penetration testers might create and exploit attacks based on the information obtained during this phase. At the attack-generating step, the information gathered is employed to create attacks on the intended web application.

The information collection process is mostly done by hand, except for the generation of attacks, which can be automated with the aid of automated attack scripts. During the response analysis phase, a tester examines the web application responses to evaluate the efficacy of the attacks and prepares a report for any vulnerabilities discovered.

Proposed Penetration Testing Methodology

The proposed Penetration methodology is shown in Fig. 2. The penetration tests are designed to map the vulnerabilities. Information gathered relates to functional requirements, functional test cases, and security risks, vulnerability knowledge base, scanning tool reports, security testing reports, and Penetration Test Coverage (PTC) metrics. Penetration tests are designed based on the information gathered. The main source of information for the penetration test design process step is based on the black-box testing method and is the set of functional scenarios/ functional test scenarios. Our objective is to develop automatic and effective attack generation algorithms from suitable models of information gathered. State models in the context of security events (vulnerabilities) are considered models of information.

Systematic Approach to Penetration Test Design

Existing methodologies lack clear guidelines to design penetration tests as a part of the penetration test process. Organizations can improve their ability to proactively identify and mitigate security risks, as well as the scalability and repeatability of security testing processes, and ultimately strengthen their overall security posture against evolving threats and vulnerabilities, by implementing automatic penetration test generation techniques. The activities for the design of the penetration test process consist of either:

1) Manually specifying necessary penetration tests by deriving them from functional test scenarios as shown in Fig. 3. Using UML sequence diagrams
   Or
2) Specifying a state model of penetration tests based on the functional test model and generating penetration tests from the model as shown in Fig. 4a and 4b

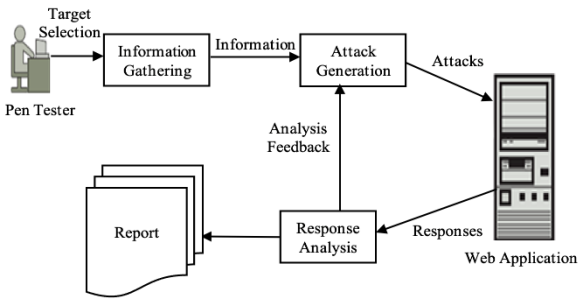

Fig. 1: Existing penetration testing methodology Halfond *et al.* (2011)

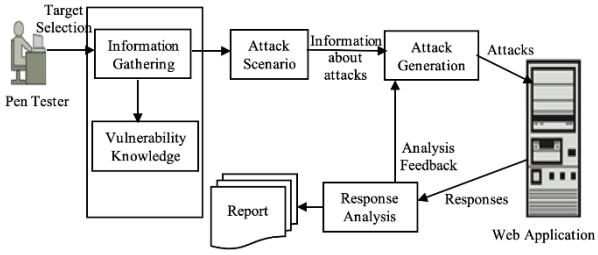

Fig. 2: Proposed penetration testing methodology

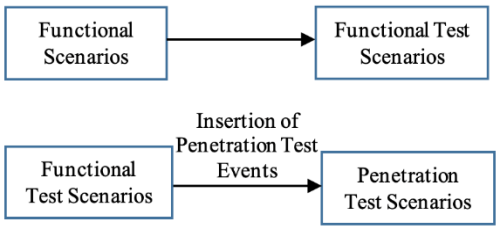

Fig. 3: Systematic approach to penetration test design using manual method

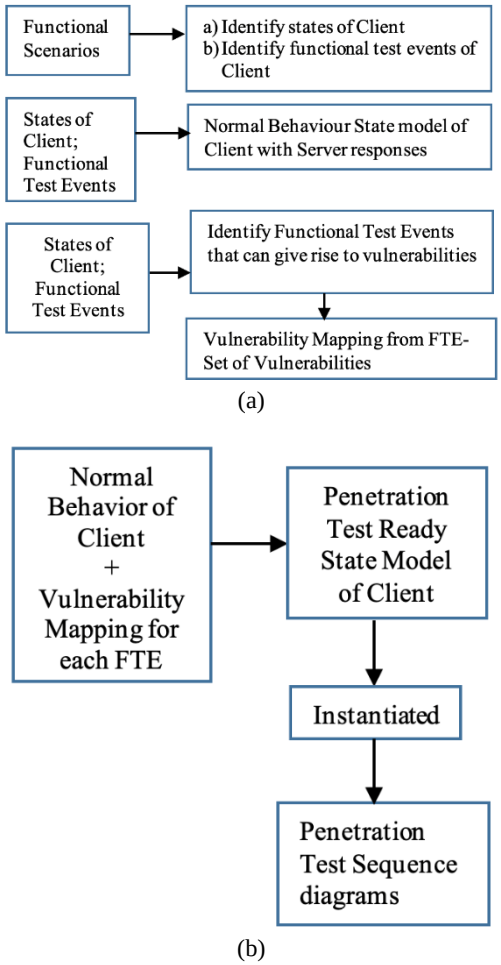

Fig. 4: Systematic approach to penetration test design using state model

Proposed Penetration Test Generation Methodology

The main source of information for the process of designing penetration tests step is the set of functional scenarios/ functional test scenarios. The activities of test process of penetration testing are affected by:

1) A change in functionality (hence, corresponding change in the functional test scenarios)
2) A change in web-based application code (driven by functionality-related changes or otherwise)
3) Updated vulnerability knowledge
4) A change in the platform on which web application is deployed

While the fact that functional test scenarios may be used as a basis for penetration test design is mentioned in Stepien *et al.* (2012) a systematic method of deriving penetration tests from functional test scenarios is lacking. Although functional test scenarios may be considered the basis for penetration test design there is no systematic methodology to develop penetration tests from functional tests Stepien *et al.* (2012). The information collected from all pertinent sources for the application that is being developed or released pertains to functional requirements, functional test cases, and security risks in the environment of the entire web application, comprising the platform on which the application runs and released as shown in Fig. 5. The information gathered impacts the design of penetration tests.

The proposed method uses the information listed below to create penetration tests that are centered on the black-box approach:

- Functional test events (functional test scenario is a sequence of functional test events)
- The set of vulnerabilities that can be exploited as each functional test event or stimulus occurs (the function vulnerabilityMapping (functional test event) is the set of vulnerabilities that can be exploited at the time the functional test event is triggered)
- Our original contribution suggested to existing penetration test processes is to introduce a step for determining vulnerability mapping (functional test event) for each and every functional test event of the web application under test

A fundamental aspect of penetration test design is creating penetration test events based on vulnerability mapping (functional test event) at each functional test event. Information gathered from architects, developers, customers (end users), field failures, and vulnerability knowledge aids in the creation of penetration test events as shown in Fig. 6.

A penetration test event is inserted at a relevant position either preceding or following each functional test event (in a functional test scenario) position based on vulnerability mapping. Individual normal or penetration test scenarios are represented as sequence diagrams.

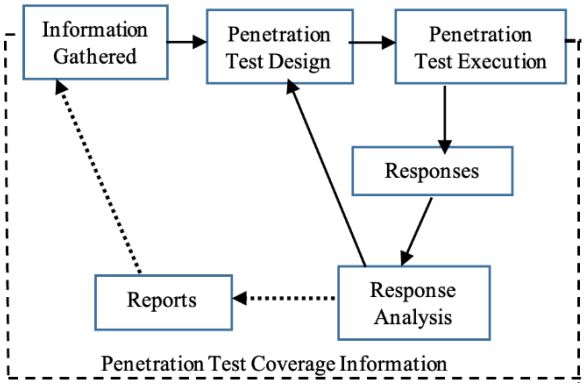

Fig. 5: Proposed penetration test generation methodology

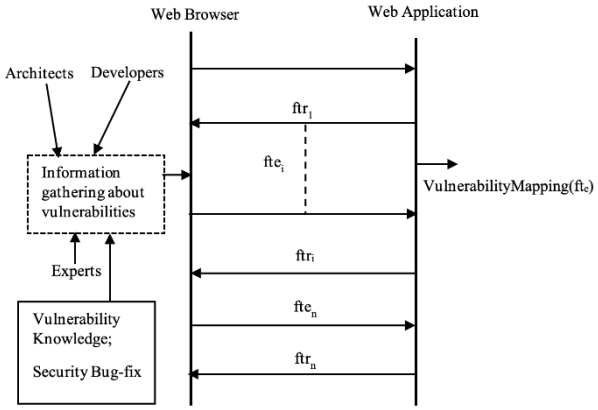

Fig. 6: Gathering information on exploitable vulnerabilities at each level of a functional test scenario

Penetration test coverage is a metric that indicates the fraction or percentage of penetration testing carried out. In this study, the Penetration Test Coverage (PTC) a part of the information gathered is proposed. PTC = 0% initially. PTC = 60% indicates that 40% of testing for penetration still needs to be done. In order to perform penetration testing, we first define PTC 1-length penetration test event sequence, or PTC 1-event, as penetration test coverage obtained by inserting only penetration test event sequences of length 1 at each functional test event position within a functional test scenario. PTC 1-event is typically set to 0% as part of the information that is gathered during penetration testing.

Functional Test Scenarios with Penetration Test Events Inserted

A functional test scenario is of the form below:

- $<fte_1, ftr_1>, <fte_2, ftr_2>, \ldots\ldots\ldots\ldots\ldots\ldots<fte_k, ftr_k>$, where fte_i is an event or stimulus in the i^{th} test step of the scenario

- ftr_i is an expected response of web application to the user's browser or client

An adequate or complete collection of functional test scenarios is a prerequisite for demonstrating the thoroughness of penetration testing. When an attack is detected using user input and cookie fields during information collection, the web application's Input Vectors (IVs) can be documented. These weaknesses are incorporated in the VulnerabilityToMapping (functional test event). Automated web crawlers are used by penetration testers to identify the IVs in the web application. A penetration test event pte is defined for a functional test scenario to be inserted at a designated functional test event position. For example, i^{th} position as in Fig. 7. To develop penetration tests as shown in Fig. 8.

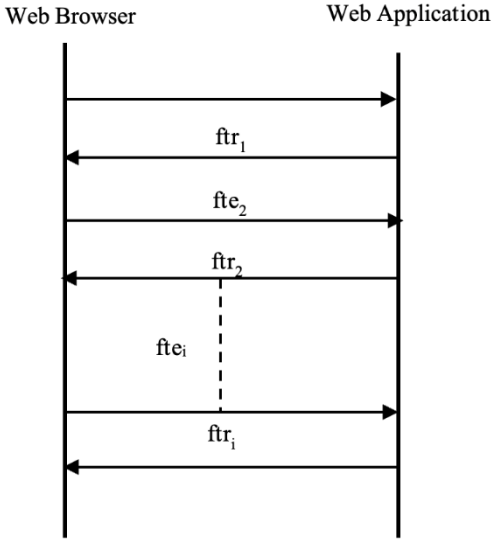

Fig. 7: Functional test scenario

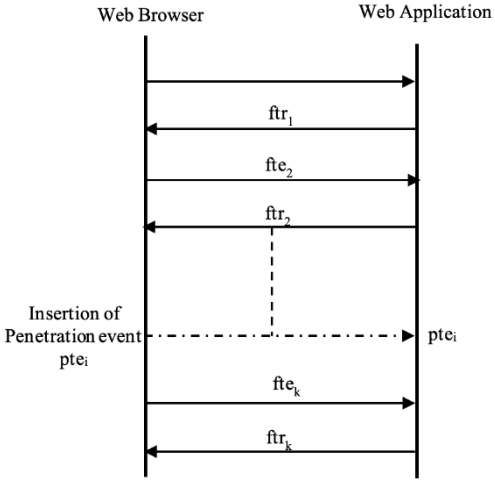

Fig. 8: Generating a penetration test by inserting a penetration test event into a functional test

Penetration testers need guidelines about pte_i to be inserted at fte_i i in [1…..k]. Hence, it is proposed that the hints or guidelines based on which pte_i is to be inserted come from vulnerability mapping information. Vulnerability mapping (fte_i) of a functional test event fte_i is the set of vulnerabilities exploited as the event fte_i occurs. As many as |Vulnerability Mapping (fte_i)| penetration tests are generated from the functional event fte_i. Each penetration test event at fte_i is created on a pte_i, i = 1. | Vulnerability mapping (fte_i)|.

Selecting a PTE (Penetration Test Event)

A pte is identified or designed by a penetration tester based on vulnerability mapping functional test scenario (fte_i) so that the vulnerability is potentially exploited. Vulnerability mapping (fte_i) > =0.

Algorithm to Derive Penetration Tests by Using UML Sequence Diagram

Algorithm 1: Penetration Path Generation Algorithm

Algorithm_Pen_Test_ Design
Input: Functional Test Scenarios and Vulnerability Knowledge
Output: Corresponding Penetration Test Scenarios

1. for i =1 to # of functional test scenarios
2. begin
3. $fts_i \leftarrow$ functional test scenario;
4. for j =1 to len (fts_i)
5. begin
6. Vulnerability Set = Vulnerability Mapping (fte_j);
7. for each V_K in the Vulnerability Set
8. begin
9. Create pte from V_K and fte_j;
10. emit fts_i after insertion of pte at fte_j as a penetration test
11. end
12. end
13. end

When the above Algorithm_Pen_Test_ Design is run, it may be necessary to note or record each pte in j € [1…….len (fts_i)] so that for insertion of a pte at fte_j, if some previous ptes are also required, (at position <j) they are also explicitly made a part of the penetration test scenario. This is expected to be required when two or more vulnerabilities arising at different functional test events need to be exploited together one after another for a combined effect to design a penetration test scenario. In this study, we confine to the insertion of only one penetration test event to create each penetration test scenario.

Design of Penetration Tests Using State Model

A case study on banking is considered and illustrated with the help of a state model and with functional test

scenarios as shown in Fig. 9. Penetration testing scenarios arise when an administrator exploits a model of the banking application, such as incorporating a new branch for banking operations. Penetration tests are formulated on functional testing. The penetration test design incorporates scenarios like "<Enter User ID, Enter Password, Enter Branch name, Enter Address, and Enter City>" derived from functional testing. The initial stage of designing a penetration test encompasses authenticating or identifying vulnerabilities at each step within a functional test scenario. By methodically examining the application's behavior in response to inputs, potential security weaknesses can be acknowledged and addressed efficiently. Additionally, considering real-world user interfaces and possible attack vectors enriches the depth of penetration testing, ensuring ample coverage of security assessments Murthy and Shilpa (2018).

Algorithm to Derive Penetration Tests from State Models

Algorithm 2: Penetration Path Generation Algorithm

"A depth-first traversal-based algorithm is used to generate functional test paths from the state model in Fig. 9.

```
genPath(S, Path)
 {
      if S is a final state
         emit(Path)
   else
      for each transition T out of S
 {
       if (not cycle (Path, T))
        {
          nextState=T.destState();
           genPath(nextState, concat(T,Path));
        }
       }
```

A test generation algorithm based on depth-first traversal has been implemented to create functional tests, as depicted in Algorithm 2. For instance, one of the functional tests derived from the state model is demonstrated in the path outlined below. For occurrence, Path 1 as shown below demonstrates one of the functional tests derived from the state model: "Path 1: State initial->Event Enter User id->State User id Entered->Event Enter Password->State Password Entered->Event Select Operation State Operation Selected (Add Branch Operation) ->Event Enter Branch Name->State Branch Name Entered->Event Enter Branch Address ->State Branch Address Entered-> Event Enter Branch City->State Branch City entered Guard condition [Entered branch does not exist already] ->State Branch Added" Murthy and Shilpa (2018).

Generation of Penetration Tests from Every Path in the Model

In the state model of the bank application, illustrated in Fig. 9, vulnerability sets are mapped to events occurring along state transitions. For example, the event "Enter User Id" is associated with the vulnerability set {SQLi, CSM} as shown in Table 1. In the formerly demonstrated finite-state machine model, each path exploits one vulnerability at each susceptible event or state transition when applying Vulnerability Length-1 Coverage. Subsequently, when considering Path 1, two instances are generated with respect to the event "Enter User Id": One targeting the SQL injection vulnerability and the other targeting the Client State Manipulation (CSM) vulnerability. Moreover, if various methods of SQL injection are attempted, multiple instances aimed at exploiting SQL injection at the "Enter User Id" event are generated. By incorporating vulnerability sets at each vulnerable state transition within Path 1, ten instances of penetration tests are generated as shown in Table 2 using the VulnerabilityMapping function. This approach ensures exhaustive coverage of potential security vulnerabilities within the bank application's functionality, enhancing its resilience against potential cyber-attacks.

Comprehensive penetration testing requires an efficient number of functional tests since they take into account all representative settings in which events or vulnerabilities could be exploited.

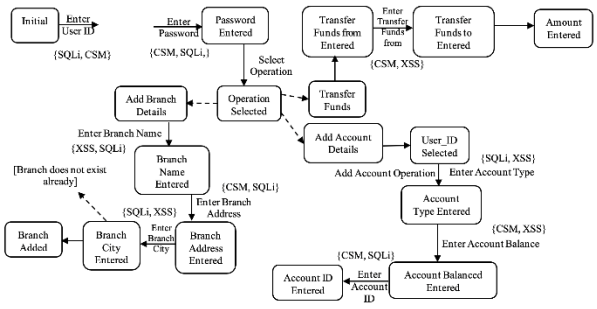

Fig. 9: Illustrating bank application using a model-based approach

Table 1: Vulnerability mapping along the functional events

Event name	Vulnerability mapping
Enter USER_ID	{SQL Injection, Client State Manipulation}
Enter Password	{Client State Manipulation, SQL Injection}
Enter Branch Name	{Cross Site Scripting, SQL Injection}
Enter Address	{Client State Manipulation, SQL Injection}
Enter City	{SQL Injection, Cross Site Scripting}

Table 2: Penetration test instances for a functional Test: Vulnerability length-1

Test ID	Enter User_ID	Enter Password	Enter Branch	Enter Address	Enter City
T1	SQLi	x	x	x	x
T2	CSM	x	x	x	x
T3	x	CSM	x	x	x
T4	x	SQLi	x	x	x
T5	x	x	XSS	x	x
T6	x	x	SQLi	x	x
T7	x	x	x	CSM	x
T8	x	x	x	SQLi	x
T9	x	x	x	x	SQLi
T10	x	x	x	x	XSS

Conclusion and Future Scope

The objective of the model-based penetration testing methodology proposed in this study is to enable automated penetration tests at a primary phase of the development of web applications. This systematic approach can be used to automatically generate attack paths in the form of penetration tests in the process of penetration testing. Based on suitable web-based application behavior models, a methodical approach and methodology for penetration test design is proposed. A framework for abstracting web application activity that takes vulnerability mapping information into account and generates automated penetration tests from state models is presented. An algorithm is designed and implemented for both the manual method using UML sequence diagrams and the automated development of penetration tests from state models.

Our original contribution involves an approach defining the behavior of the web application by constructing comprehensive state models. These models incorporate different inputs, outputs, states, transitions, and user and external system interactions. A structured basis for additional analysis by using methods like as state charts, Petri nets, or finite state machines to represent the dynamic behavior of the application is created. We have designed and implemented algorithms for both automatic and manual penetration test generation in order to implement our methodology. Using model checking, these algorithms methodically extract penetration tests from the state models and examine the state space of the application. To validate the efficacy of our approach, a case study demonstrating bank application is presented. This case study illustrates how the state model is constructed, how vulnerabilities are mapped onto the model, and how test cases are derived both automatically and manually. By executing these tests against real-world web applications, the methodology's ability to uncover and mitigate security vulnerabilities in a structured, systematic manner is demonstrated. Coverage can improve the methodology's efficacy. In this regard penetration test occurrences for a functional test vulnerability Length-1 coverage is proposed which is an original research contribution to the field of penetration testing.

The future scope of the work will involve applying the proposed automated methodology for generating penetration tests to other classes of vulnerabilities, such as session management and application management vulnerabilities, such as broken authentication and Cross-Site Request Forgery (CSRF). Furthermore, the developed algorithms and techniques will support the practical application of our methodology. These tools support constructing and visualizing state models, identifying vulnerabilities, generating test cases, and executing penetration tests. By seamlessly integrating with existing development and testing workflows, research aims to promote the adoption and application of the proposed methodology by security practitioners and software developers alike.

Acknowledgment

I express my gratitude to Dr. T. P. Pushphavathi and Dr. P. V. R. Murthy, my fellow authors, for their invaluable guidance and mentoring. Their domain expertise in security was really beneficial to me as I completed my task. I am grateful to Paladion Networks Pvt. Ltd for providing the web application needed to complete our research.

Funding Information

There was no funding obtained to help in the writing of this manuscript.

Author's Contributions

Shilpa R. G.: Contributed to the written paper and outcomes.

Pushphavathi T. P. and Murthy P. V. R.: Recommendations were made after the entire manuscript was evaluated.

Ethics

By signing this, I, Ms. Shilpa R.G., verify that the following is true for this manuscript:

1) Nothing that has been published before is original to the writers and is their own creation
2) No other publications are currently considering publishing the paper
3) The work precisely and fully signifies the authors' own investigation
4) The findings are appropriately contextualized within the body of previous and current research
5) Every author has personally and actively contributed significantly to the work that resulted in the publication and they will accept public accountability for its contents

References

Awang, N. F., & Manaf, A. A. (2013). Detecting Vulnerabilities in Web Applications Using Automated Black Box and Manual Penetration Testing. In *Advances in Security of Information and Communication Networks* (Vol. 381, pp. 230–239). https://doi.org/10.1007/978-3-642-40597-6_20

Bacudio, A. G., Yuan, X., Bill Chu, B. T., & Jones, M. (2011). An Overview of Penetration Testing. *International Journal of Network Security & Its Applications*, *3*(6), 19–38. https://doi.org/10.5121/ijnsa.2011.3602

Benikhlef, I., Wang, C., & Gulomjon, S. (2016). Mutation Based SQL Injection Test Cases Generation for the Web Based Application Vulnerability Testing. *Proceedings of the 2nd International Conference on Electronics, Network and Computer Engineering (ICENCE 2016)*, 546–551. https://doi.org/10.2991/icence-16.2016.104

Casola, V., De Benedictis, A., Mazzocca, C., & Orbinato, V. (2024). Secure software development and testing: A model-based methodology. *Computers & Security*, *137*, 103639. https://doi.org/10.1016/j.cose.2023.103639

Choiriyah, A., & Qomariasih, N. (2023). Security Analysis on Websites Belonging to the Health Service Districts in Indonesia Based on the Open Web Application Security Project (OWASP) Top 10 2021. *2023 International Conference on Information Technology and Computing (ICITCOM)*, 267–272. https://doi.org/10.1109/icitcom60176.2023.10442816

Felderer, M., Agreiter, B., Zech, P., & Breu, R. (2011). *A Classification for model-based security testing*. 109–114.

Felderer, M., Zech, P., Breu, R., Büchler, M., & Pretschner, A. (2016). Model-based security testing: a taxonomy and systematic classification. *Software Testing, Verification and Reliability*, *26*(2), 119–148. https://doi.org/10.1002/stvr.1580

Garousi, V., Keleş, A. B., Balaman, Y., Güler, Z. Ö., & Arcuri, A. (2021). Model-based testing in practice: An experience report from the web applications domain. *Journal of Systems and Software*, *180*, 111032. https://doi.org/10.1016/j.jss.2021.111032

Halfond, W. G. J., Choudhary, S. R., & Orso, A. (2011). Improving penetration testing through static and dynamic analysis. *Software Testing, Verification and Reliability*, *21*(3), 195–214. https://doi.org/10.1002/stvr.450

Lebeau, F., Legeard, B., Peureux, F., & Vernotte, A. (2013). Model-Based Vulnerability Testing for Web Applications. *2013 IEEE 6th International Conference on Software Testing, Verification and Validation Workshops*, 445–452. https://doi.org/10.1109/icstw.2013.58

Li, X., & Xue, Y. (2014). A survey on server-side approaches to securing web applications. *ACM Computing Surveys*, *46*(4), 1–29. https://doi.org/10.1145/2541315

Lonetti, F., Bertolino, A., & Di Giandomenico, F. (2023). Model-based security testing in IoT systems: A Rapid Review. *Information and Software Technology*, *164*, 107326. https://doi.org/10.1016/j.infsof.2023.107326

Lunkeit, A., & Schieferdecker, I. (2018). Model-Based Security Testing - Deriving Test Models from Artefacts of Security Engineering. *2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 244–251. https://doi.org/10.1109/icstw.2018.00056

Murthy, P. V. R., & Shilpa, R. G. (2018). Vulnerability Coverage Criteria for Security Testing of Web Applications. *2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 489–494. https://doi.org/10.1109/icacci.2018.8554656

Peleska, J., Brauer, J., & Huang, W. (2018). Model-Based Testing for Avionic Systems Proven Benefits and Further Challenges. In *Leveraging Applications of Formal Methods, Verification and Validation. Industrial Practice* (Vol. *11247*, pp. 82–103). Springer. https://doi.org/10.1007/978-3-030-03427-6_11

Peroli, M., De Meo, F., Viganò, L., & Guardini, D. (2018). MobSTer: A model-based security testing framework for web applications. *Software Testing, Verification and Reliability*, *28*(8), e1685. https://doi.org/10.1002/stvr.1685

Schmidt, A., Durak, U., & Pawletta, T. (2016). Model-based testing methodology using system entity structures for MATLAB/Simulink models. *SIMULATION*, *92*(8), 729–746. https://doi.org/10.1177/0037549716656791

Sommer, F., Kriesten, R., & Kargl, F. (2021). Model-Based Security Testing of Vehicle Networks. *2021 International Conference on Computational Science and Computational Intelligence (CSCI)*, 685–691. https://doi.org/10.1109/csci54926.2021.00179

Sommer, F., Kriesten, R., & Kargl, F. (2023). Survey of Model-Based Security Testing Approaches in the Automotive Domain. *IEEE Access*, *11*, 55474–55514. https://doi.org/10.1109/access.2023.3282176

Stepien, B., Peyton, L., & Xiong, P. (2012). Using TTCN-3 as a modeling language for web penetration testing. *2012 IEEE International Conference on Industrial Technology*, 674–681. https://doi.org/10.1109/icit.2012.6210016

Xiong, P., & Peyton, L. (2010). A model-driven penetration test framework for Web applications. *2010 Eighth International Conference on Privacy, Security and Trust*, 173–180. https://doi.org/10.1109/pst.2010.5593250