Journal of Electronic Research and Application, 2025, Volume 9, Issue 4
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Research on Software Security Testing Mode Based on Open Network Environment

Xiaoqing Jia*

New York Institute of Technology, New York, 10023 United States

**Author to whom correspondence should be addressed.*

Abstract: Starting with the goal and significance of software security testing, this paper introduces the main methods of software security testing in the open network environment, including formal security testing, white box testing, fuzzy testing, model testing, and fault injection testing. A software security testing method based on a security target model is proposed. This paper provides new ideas for software security testing, better adapts to the open network environment, improves the efficiency and quality of testing, and builds a good software application environment.

Keywords: Network environment; Software; Safety performance; Test method; Safety protection

Online publication: August 7, 2025

1. Introduction

With the wide application of 5G, cloud computing, and Internet of Things technologies, software systems are increasingly embedded in open network environments, and their operating boundaries are further extended to the world. An open network environment has the characteristics of node heterogeneity, access dynamics, and attack universality, which puts forward new requirements for security testing. On the one hand, software needs to deal with continuously evolving network attack means, such as APT attack and zero-day vulnerability utilization; On the other hand, we should actively prevent the risk of data leakage and out of control permissions, and provide technical support for third-party services. Relevant surveys show that security events caused by application layer vulnerabilities account for more than 60% in the open network environment, exposing the lack of existing testing mechanisms [1]. Based on this, this study focuses on the uniqueness of the open network environment, discusses the software security testing methods and optimization and improvement measures, and provides empirical reference for the safe and stable operation of software.

2. Objectives and significance of software security testing

2.1. Test objectives

The objectives of software security testing are: (1) During the development of software, there may be security vulnerabilities due to non-standard code writing, logic design defects, third-party component vulnerabilities, and other reasons. The primary goal of security testing is to use relevant testing techniques and methods to find out these vulnerabilities and take measures to repair them. (2) To ensure software security, developers usually develop a series of security policies and mechanisms, such as user identity authentication, access control, data encryption, and so on. The software security test can verify whether these policies and mechanisms are effective. (3) In different industries and fields, there are corresponding software security standards and regulations, such as the Payment Card Industry Data Security Standard in the financial industry, and the Health Insurance Circulation and Liability Act in the medical industry. Software security testing should ensure that the software strictly complies with these standards and regulations in the process of design, development, and operation, meets compliance requirements, and enhances users' trust in the software.

2.2. Significance

First, protect user data security. User data is one of the most valuable assets in the software system, including personal identity information, contact information, financial data, and so on. Once these data are leaked, it will bring serious losses to users. Software security testing ensures the security of data throughout the whole life cycle through strict detection of data storage, transmission, and processing.

Second, ensure the stable operation of key businesses. In financial, medical, transportation, and other fields, software systems are responsible for the operation of key businesses. Once security problems occur, they may lead to business interruption, service paralysis, or even serious adverse consequences. Software security testing can ensure that these key business systems have sufficient security protection capabilities, effectively resist various security threats, ensure business continuity and stability, and provide strong support for the normal operation of society [2].

Third, promote the healthy development of the software industry. With the rapid development of the software industry, software security issues are increasingly of concern. Strengthening software security testing will help promote the whole industry to attach importance to software security and encourage enterprises to increase investment in technology research and development and talent training. At the same time, by constantly summarizing the experience and methods of safety testing, it can improve the overall safety technology level of the industry, form a good safety ecological environment, and promote the healthy development of the software industry.

3. Main methods of software security testing in an open network environment

3.1. Formal security test

Formal security testing uses mathematical logic and formal methods to verify whether the software system meets the preset security attributes, such as confidentiality, integrity, and availability, by establishing an abstract mathematical model and using theorem proof or model detection. The core of this test method is to transform the security requirements into standardized and formal rules, thus proving the consistency of system behavior and rules. In an open network environment, the formal security test method is particularly suitable for verifying the logical integrity of key components such as communication protocols and identity authentication mechanisms to

resist network threats such as man-in-the-middle attacks and replay attacks.

The specific test methods include: (1) theorem proving. Using higher-order logic such as Isabelle/HOL, Coq, manually or semi-automatically deduce the mathematical proof that the system conforms to the security rules, which is applicable to the formal verification of encryption algorithms or zero trust architecture. (2) Formal analysis of protocols. For TLS and OAuth 2.0 network protocols, Pi calculus or communication sequential process is used to model the message interaction process and detect session hijacking or privilege escalation vulnerabilities. In the open network environment, the traditional formal model needs to be extended in the testing work, and timed automata are introduced to deal with delay tolerance, or the probability model is used to quantify the success rate of uncertain attacks.

3.2. White box test

White box testing is used to identify potential vulnerabilities by combining static analysis with dynamic execution for software internal structure, code logic, and data flow [3]. The core of this test method is to deeply understand the program execution path, covering code branches, boundary conditions, and exception handling mechanisms. In an open network environment, white box testing is particularly applicable to security flaws caused by cross-service calls, third-party API integration, or configuration errors.

The specific test methods include: (1) Static code analysis. Detect code-level vulnerabilities such as buffer overflow and SQL injection through lexical analysis, control flow graph construction, and data flow analysis. Common tools such as Checkmark and Fortify can support cross-language rule base matching. (2) Dynamic symbol execution. Combined with specific execution and symbolic variables, high-coverage test cases are generated to explore deep code paths, typically represented by the KLEE framework. (3) Stain analysis. Trace the propagation path of external input in the program to identify sensitive operations caused by unverified data, such as file writing and system command execution. In the open network environment, for the microservice architecture in the test work, it is necessary to build a distributed data flow diagram to track the leakage of cross-node sensitive information; At the same time, in combination with the container environment, analyze whether the relevant configuration and the proxy security policy are compliant.

3.3. Fuzzy test

Fuzzy testing involves inputting a large amount of random data to the target program, triggering the abnormal behavior of the system, and relying on the input generation strategy and coverage feedback mechanism to find potential vulnerabilities. The operation of this test method is relatively simple: the first step is to prepare a correct file inserted into the program; The second step is to replace the local content of the file with random data; Step three is to open the file with a program; The fourth step is to observe the impact and damage. In an open network environment, fuzzy testing needs to cover a variety of protocols and data formats to deal with the complexity of interaction between heterogeneous devices and services.

The specific test methods include: (1) Based on generation. Structured malformed input is constructed according to protocol specifications or syntax templates, which are suitable for API or communication protocol testing. Common testing tools such as Boofuzzy. (2) Based on variation. Perform bit flipping, field truncation, and other operations on legal samples to quickly generate test cases. The common test tool is AFL, which is suitable for file parser testing. (3) Coverage guidance. The pile insertion technology is used to monitor the code coverage and dynamically optimize the input generation direction, such as LibFuzzy combined with AFL++, which supports

multi-threaded distributed execution. In the open network environment, the network protocol state machine model should be integrated in the fuzzy test to ensure that the test process covers protocol handshake, encryption negotiation, and other state migration. The intelligent fuzzy strategy is adopted to further improve the discovery efficiency of unknown protocol vulnerabilities and the accuracy of test results.

3.4. Model testing

Model testing is to build an abstract behavior model of the software system and generate test cases based on the model to verify whether the system conforms to the expected safety behavior. The core of the test is to simplify the complex system and expose design defects by traversing the model state space. Classical test models include V model, W model, X model, and H model [4]. Taking the W model as an example, it is an upgrade of the V model. The test covers the entire development cycle of the software. The test object is not only the program, but also the requirements, functions, and design, which is conducive to finding problems as soon as possible. In an open network environment, model testing is applicable to verifying the consistency and fault tolerance of distributed consensus algorithms, blockchain smart contracts, and other scenarios.

The specific test methods include: (1) state machine modeling. Use UML state diagram or the TLA+ language to describe system state transition logic and generate a test sequence covering all migration paths. (2) Model-driven test generation. Automatically export test cases and predictions based on models, and common tools such as GraphWalker. (3) Probability model validation. In the face of a random network environment, a random model is used to evaluate the security attributes of the system under uncertain conditions. In the open network environment, the spatio-temporal model needs to be introduced into the model test to describe the relationship between the geographical location of nodes and communication delay; Combined with the game theory model, it simulates the strategic interaction between the attacker and the defender, and quantifies the effectiveness of the security defense mechanism.

3.5. Fault injection test

Fault injection test is to artificially introduce a hardware fault, network anomaly, or software error, and evaluate the fault tolerance and recovery mechanism of the software system under abnormal conditions. This testing method can actively discover potential weaknesses in software that cannot be detected by traditional testing methods, so that testers can observe the behavior of software under adverse conditions, which helps to achieve higher code coverage and ensure that the software can effectively deal with unexpected situations without catastrophic consequences. In an open network environment, fault injection testing is suitable for simulating complex scenarios such as distributed node failure, message loss, and Byzantine failure.

The specific test methods include: (1) Code-based fault injection. Tamper the return value of a function or the status of a variable through the stub insertion or hook technology. Common tools such as LLVM Fault Injector. (2) Interface-based fault injection. Intercept network communication or API calls, inject delay, error codes, or malformed data, and commonly used tools such as Gremlin. (3) Environment-based fault injection. Simulate resource exhaustion, clock drift, or network partitioning. In the open network environment, the fault injection test needs to build a multi-dimensional fault model library, covering the energy fluctuations of edge computing nodes, wireless channel interference, and other physical layer anomalies, and evaluate the system's resilience under continuous attacks [5].

4. Software security testing methods based on the security target model

4.1. Safety target model

In 1996, six countries, including the United States and Canada, jointly issued version 1.0 of the Common Criteria for Information Technology Security Assessment CC (ISO/IEC 15408), marking the unification of international information security assessment criteria. In 2001, China officially promulgated the CC based national standard Information Technology Security Technology Information Technology Security Assessment Criteria (GB/T 18336-2001), which provides guidance for China's information security assessment and certification work. This criterion divides the security protection level of computer information systems into five levels, which becomes the general evaluation basis of information system security evaluation and also provides a basis for establishing the security target model.

4.2. Test requirement analysis

For typical software products, risk identification is carried out based on the protected objects and different boundaries, which can be summarized into six categories of security objectives, namely management protection, user data protection, user boundary protection, password system boundary protection, computing environment security protection, and physical security level one security protection. The first-level security objectives are refined, and each security objective corresponds to multiple security attributes to further determine the corresponding security threats. After identifying security threats, quantify them from five dimensions of reproducibility, availability, discoverability, affected users, and potential losses, calculate the risk value of each threat, determine the priority, select necessary test items, and propose the security test framework and specific content.

4.3. Test contents and methods

(1) Data entity exchange authentication

For the data entity exchange authentication item, the security test content includes an authentication failure processing mechanism, entity authorization definition, user authentication, and the correctness of entity subject and behavior binding. The test method is to simulate the number of user's identity authentication attempts, and verify the user's identity authentication failure processing mechanism; The function covers the definition of the access permission of the test device to the user, and tests whether the main body is consistent with the behavior binding.

(2) User business boundary protection

For user business boundary protection items, take the controlled access of network boundary protection control as an example, and the security test content includes the correctness of controlled access rules and the availability of controlled access functions. The test method is code analysis and simulation coverage, to verify whether the access control mechanism of network boundary devices for accessing users is consistent with the design requirements, and to analyze and verify the effectiveness of failure protection and state recovery. In addition, the function simulation verifies the status synchronization and time identification validity, and the cycle function covers the reliability of automatic script construction to verify the anti-repudiation function.

(3) User application data protection

The user application data protection item includes such sub-items as service anti-repudiation, user

application data transmission, and user application data storage. Taking the security test of user application data transmission as an example, the test method is code analysis and function coverage, verification of transmission data protection access control, identification of data source and producer, protection of residual information of transmission data, correctness of data integrity and confidentiality mechanism, and verification of anti-repudiation function reliability by automated script construction.

(4) Manage data security

For the management data security protection item, take the equipment management data generation verification as an example. The test method is to verify whether the equipment management item covers the design or specification; Whether the equipment management data is correct and effective; The authority of management data generation rule management is verified through function simulation, and the correctness of management data generation rule management authorization is tested.

(5) Operation environment safety protection

For the security protection items of the running environment, take vulnerability scanning verification as an example; it is necessary to verify whether the operating system, applications, and other components of the tested device have known security vulnerabilities. The test method is to use the Lvmeng network security vulnerability scanning system to scan the network access equipment to ensure the network accessibility of the system and the tested equipment; Perform system scanning on the operating system, application program, and other components of the tested device.

5. Conclusion

With the rapid development of information technology, software architecture has become more complex, and functional modules have increased. At the same time, security flaws and vulnerabilities are also increasing, threatening the stable operation of software. This paper introduces the main ways of software security testing under the open network environment, and prompts testers to change their traditional thinking and adopt the testing method based on the security goal model, so as to further improve the comprehensiveness and accuracy of the test results and protect the reliable operation of the software system.

Disclosure statement

The author declares no conflict of interest.

References

[1] Gulay GH, Bedir T, Cagatay C, et al., 2024, Test Suite Assessment of Safety-Critical Systems Using Safety Tactics and Fault-Based Mutation Testing. Cluster Computing, 27(4): 5377–5401.

[2] Watson RNM, Chisnall D, Clarke J, et al., 2024, CHERI: Hardware-Enabled C/C++ Memory Protection at Scale. IEEE Security & Privacy, 22(4): 50–61.

[3] Rani SA, Akila C, Raja SP, 2024, Guided Intelligent Hyper-Heuristic Algorithm for Critical Software Application Testing Satisfying Multiple Coverage Criteria. J Circuit Syst Comp, 33(2): 1–29.

[4] Pradhan V, Dhar J, Kumar A, 2023, Testing Coverage-Based Software Reliability Growth Model Considering Uncertainty of Operating Environment. Systems Engineering, 26(4): 449–462.

[5] Li L, Tian S, Zhou W, et al., 2023, Application of Driving Simulators in the Validation Test for Vehicle Active Safety System. Int J Crashworthiness, 28(1/2): 159–169.