Research on Computer Software Security Testing Techniques and Applications

Xinyang Jia
Institute of Problem Solving, Xi'an FanYi University, Xi'an, Shaanxi, China

Abstract: Computer software security testing is a critical step in ensuring the security of computer systems. This research focuses on the techniques and applications of computer software security testing. Firstly, an analysis is conducted on three common testing methods: black-box testing, white-box testing, and grey-box testing. Next, the analysis and evaluation of security testing technology based on vulnerability mining, static analysis, and dynamic analysis are discussed. Lastly, the application practice of computer software security testing technology in web applications and mobile applications is explored. This research provides insight and guidance for computer software security testing.

Keywords: Computer Software Security Testing; Black-Box Testing; White-Box Testing; Grey-Box Testing; Vulnerability Mining; Static Analysis; Dynamic Analysis; Web Applications; Mobile Applications

1. Overview of Computer Software Security Test Technology

1.1 Black-Box Test

Black box teBlack box testing, also known as functional testing, is a method of testing based on software functionality, regardless of the internal structure and implementation details of the software. It enters a set of test cases to test whether the software functions expected, whether it can correctly respond to the input and generate the corresponding output, and whether it has good fault tolerance and robustness. The objective is to verify that the software system is operating properly as required in the Requirements specification. It focuses on the testing of aspects of the software's functionality, usability, and user experience. By simulating real usage scenarios and typical user behavior, black-box tests can detect potential functional defects and errors, and fix and improve the software. In black box tests, common methods include functional, performance, stress, and safety tests.

1.2 White-Box Test

White box teWhite box test, also known as structure test or transparent box test, is a test method based on an understanding of the internal structure and programming code of the software. Unlike black box testing, white box testing requires testers to have an understanding of the internal implementation details of the software, including code logic, process control, data structure, etc. By analyzing the software code, testers can compile test cases to test whether the software code meets expectations, works correctly, and eliminate potential defects or vulnerabilities. The goal is to verify that the code logic of the software is accurate, complete, and efficient. It focuses on the interaction and data flow between the modules, and the coverage and quality of the code. By detecting code errors, logic flaws, and potential security vulnerabilities, white-box testing can help developers fix problems and improve the reliability and security of the software.

1.3 Grey-Box Test

The grey box test, also known as the translucent box test, is a test method between the black box test and the white box test. The grey box test considers both the external behavior and function of the software and the internal structure and implementation details of the software. In ash box testing, the tester can usually access some of the internal structure of the software, such as databases, log files, etc., to compile test cases for testing. The goal is to consider the internal and external factors of the software to verify whether the function, performance and security of the software meet expectations[1]. It focuses on whether the function of the software is correct, and whether the logic of the software is consistent, whether the performance is optimized, and whether

there are potential security vulnerabilities. By combining the advantages of black box testing and white box testing, grey box testing can provide more comprehensive and in-depth test coverage and discover more potential problems.

2. Analysis and Evaluation of Computer Software Security Test Technology

2.1 Security Testing Technology based on Vulnerability Mining

2.1.1. The Working Principle

The security testing technology based on vulnerability mining is mainly based on two basic principles: fuzzy testing and vulnerability exploitation. Fuzzy testing is to trigger a potential vulnerability by entering abnormal or unexpected input into a program; the exploit is to execute malicious code or gain unauthorized access.

2.1.2. Advantages and disadvantages

Security testing technology based on vulnerability mining has the following advantages and disadvantages:

merit:

1.Find the hidden security vulnerabilities: The security testing technology based on vulnerability mining can find the hidden security vulnerabilities in the software system, including unauthorized access, buffer overflow, SQL injection, etc. This approach evaluates the security of the software system from the attacker's perspective.

2.Wide coverage: The security testing technology based on vulnerability mining can carry out comprehensive penetration testing and vulnerability scanning on the software system, with a wide coverage. By simulating various attack methods and attack scenarios, various types of security vulnerabilities can be discovered.

3.Improve the security of the software system: By discovering and repairing the security vulnerabilities, the security testing technology based on vulnerability mining can help improve the security of the software system. Potential attacks and data leaks can be prevented by timely fixing the vulnerabilities.

shortcoming:

1.Low efficiency of vulnerability mining: The security testing technology based on vulnerability mining requires a lot of time and resources, especially for complex software systems and large-scale testing. Both fuzzy testing and vulnerability exploitation are a trial-and-error process that may require multiple attempts to discover effective vulnerabilities.

2.Difficulty in mining: The security testing technology based on vulnerability mining has high technical requirements for testers. Tsters need to have certain security knowledge and skills to be able to understand the principles and methods of vulnerability mining and to skillfully use relevant tools and techniques.

3.Not guarantee the discovery of all vulnerabilities: the security testing technology based on vulnerability mining does not guarantee the discovery of all security vulnerabilities. Due to the diversity and complexity of vulnerabilities, some vulnerabilities may not be discovered through this method, or require more specialized testing methods and techniques.

2.1.3. Common tools and techniques

There are many common tools and techniques available to select and use in vulnerability mining based security testing as follows:

1.Fuzzy test tool: Fuzzy test tool is used to automate the fuzzy test, such as AFL (American Fuzzy Lop), Peach Fuzzer, etc. These tools are able to automatically generate large numbers of abnormal and unintended inputs to trigger potential vulnerabilities.

2.Vulnerability utilization tools: Vulnerability exploitation tools are used to simulate the attacks of attackers on the software system. For example, Metasploit is a popular vulnerability exploitation tool that provides a large number of vulnerability modules and attack scripts that can be used to test system vulnerabilities.

3.Security audit tools: Security audit tools are used to conduct static and dynamic analysis of the software system to find potential security vulnerabilities. For example, static code analysis tools can analyze potential vulnerabilities in the source code, and dynamic analysis tools can simulate attack behavior in the actual running environment.

4.Vulnerability database: Vulnerability database can provide the vulnerability related information and repair suggestions. For example, CVE (Common Vulnerabilities and Exposures) is a widely used vulnerability database that helps testers understand known vulnerabilities and corresponding fixes.

5.Vulnerability mining technology: Vulnerability mining technology includes static and dynamic analysis, symbol execution, fuzzy

testing, reverse engineering, etc. These technologies can help testers find potential vulnerabilities in their software systems, such as by analyzing the source code of the program and simulating the behavior of the attackers.

2.2 Safety Test Technology based on Static analysis

2.2.1. The Working Principle

The safety test technology based on static analysis mainly relies on the analysis and detection of the program code, and finds out the potential safety problems by modeling and verifying the static features of the program.[2]

2.2.2. Advantages and disadvantages

The safety testing technique based on static analysis has the following advantages and disadvantages:

merit:

1.Comprehensive coverage: The security test technology based on static analysis can analyze the source code of the whole program, with a wide coverage. By analyzing the static features of the program, the potential defects and vulnerabilities in the code can be discovered.

2.In-depth analysis: The safety test technology based on static analysis can conduct an in-depth analysis of the logic and data flow of the program. By checking the flow and change of the data, and the program, you can discover hidden security issues in the program.

3.Discovery of hidden vulnerabilities: The security testing technology based on static analysis can discover hidden vulnerabilities in programs, that is, those vulnerabilities that are not easy to be discovered by traditional testing methods. The detailed analysis of the program code can reveal the potential problems in the code and thus improve the security of the software system.

shortcoming:

1.Long analysis time: The safety test technology based on static analysis requires detailed analysis of the source code of the program, for a long time. Especially for large and complex software systems, the analysis time may be longer.

2.High false positive rate: The safety test technology based on static analysis may produce a large number of false positives in the analysis process, namely, falsely marking the normal code as a potential security problem. This can lead to additional workload and complexity, as well as further analysis and judgment of the testers.

3.Difficult to verify the repair effect: The safety test technology based on static analysis is often difficult to directly verify the repair effect after the safety problem is found. Because of the analysis of static features based on the code, the actual environment and attack scenario of the runtime cannot be simulated, so the repair effect needs to be tested twice.

2.2.3. Common tools and techniques

There are many common tools and techniques available to choose and use in safety testing based on static analysis, as follows:

1.Static code analysis tool: The static code analysis tool is used to conduct a static analysis of the source code of the program and to find potential security problems. For example, tools such as FindBugs, PMD, and Coverity can help testers find potential flaws and vulnerabilities in their code.

2.Data flow analysis tool: Data flow analysis tool is used to analyze the data flow of the program and discover the data transmission and processing methods that may lead to security problems. For example, tools such as CodeSonar and Fortify can help testers detect security issues such as sensitive data breaches and unverified user input.

3.Control flow analysis tool: The control flow analysis tool is used to analyze the control process of the program and find logical errors and vulnerabilities that may lead to security problems. For example, tools such as Coverity and CodeSonar can help testers identify security issues such as incorrect validation, access control errors, and logic vulnerabilities.

4.Model Verification tool: The model validation tool is used to verify the model of the program and check whether it meets the specific safety nature. For example, tools such as ESBMC, CBMC can help testers verify the security policies and access control logic of programs.

2.3 Safety Testing Technology based on Dynamic Analysis

2.3.1. The Working Principle

The security test technology based on dynamic analysis mainly uses the monitoring and analyzing the behavior of the program at the running time to discover the potential security problems.

2.3.2. Advantages and disadvantages

The safety testing technique based on dynamic

analysis has the following advantages and disadvantages:

merit:

1.High test efficiency: The security test technology based on dynamic analysis can quickly test the program, and find out potential security problems by monitoring the behavior of the program during operation. Compared with methods based on static analysis, dynamic analysis techniques can find vulnerabilities and errors faster.

2.Wide coverage: The security testing technology based on dynamic analysis can cover the actual execution process of the program, including input processing, function calling, memory access, etc. By monitoring and analyzing the behavior of the program at runtime, security problems caused by dynamic behavior can be found.

3.Finding unknown vulnerabilities: Security testing technology based on dynamic analysis can discover unknown security problems, that is, those vulnerabilities that are not easy to be discovered by traditional testing methods. Methods such as fuzzy testing and symbol execution can probe the behavior of programs when dealing with unexpected inputs and complex logic.

shortcoming:

1.High requirements of the test environment: the safety test technology based on dynamic analysis has higher requirements on the test environment. Tsters need to be able to simulate the real operating environment and monitor and analyze the execution process. This may require a certain hardware and software configuration.

2.Reliability of test results: The safety test technology based on dynamic analysis may produce certain false positives and underreports in the analysis process, that is, the normal behavior is wrongly marked as a potential safety problem, or fails to find the real safety problems. Tsters need to conduct further analysis and verification of the test results to accurately determine whether there is a safety problem.

3.Impact on program performance: Security testing technology based on dynamic analysis will have a certain impact on the performance of the program. During testing, the runtime behavior of the program needs to be monitored and analyzed, which may cause the program to run slower or consume more resources.

2.3.3. Common tools and techniques

There are many common tools and techniques available to choose and use in safety testing based on dynamic analysis, as follows:

1.Fuzzy test tool: Fuzzy test tool is used to automate the fuzzy test, such as AFL (American Fuzzy Lop), Peach Fuzzer, etc. These tools are able to automatically generate large numbers of abnormal and unintended inputs to trigger potential vulnerabilities.

2.Symbol execution tool: Symbol execution tool is used to sign the symbolic variables of the program to execute various path and conditional branches of the program. For example, tools like KLEE, SAGE can help testers in symbol execution and vulnerability discovery.

3.Dynamic analysis tools: Dynamic analysis tools are used to monitor and analyze the behavior of the program at runtime. For example, tools such as Valgrind and DynamoRIO can help testers detect security issues such as memory access errors and resource leaks.

4.Hybrid execution tools: Hybrid execution tools are used to find potential vulnerabilities and errors by comprehensively considering the execution of symbol execution and specific inputs. For example, tools such as Angr, Triton can help testers with mixed execution and safety analysis.

3. Application Practice of Computer Software Security Test Technology

3.1 Safety Testing for Web Applications

In practical applications, many enterprises and organizations will safely test their Web applications to ensure their security and reliability. The methods of safety test can be divided into manual test and automatic test.

Manual testing means when a professional tester discovers security vulnerabilities to the system through simulated attacks. The tester will test the system safely using various methods and technologies according to the functions and requirements of the system. For example, a tester can try to use malicious data for input to test if the system verified and filtered the input. Manual testing can more flexibly detect security problems in the system, but it requires more time and human resources.

Automatic testing is using automatic tools to test the safety performance of the system. Automated test tools can quickly discover

potential security problems in the system by simulating attacks, scanning vulnerabilities, and checking configurations. Commonly used Web application security testing tools include Selenium, Burp Suite, etc[3]. These tools are characterized by high efficiency, accuracy and repeatable, which can improve test efficiency and reliability of test results.

Whether manual or automated testing, security testing for Web applications needs to follow certain testing procedures and methods. The usual test process includes requirements analysis, test plan formulation, test case design, test execution, test results analysis and report, etc. Tsters need to choose appropriate test methods and technologies according to the characteristics and requirements of the system, record the test process and test results, and timely report and repair the safety problems found.

3.2 Security Testing of Mobile Applications

There are two main safety test methods for mobile applications: manual test and automatic test.

Manual testing refers to the discovery of security vulnerabilities in the system through simulated attacks, which requires professional tester to test. In the manual test of mobile applications, the tester will test all aspects of the application, including input verification, permission control, data transmission, data storage, security configuration, etc.

Automated testing is to test the security performance of mobile applications by using automated testing tools. Automated testing tools can simulate attacks and discover potential security issues in applications, such as vulnerabilities, weaknesses, insecure configurations, etc. Commonly used automated testing tools for mobile applications include Appium, Monkey, etc.

In practice, many organizations will test their mobile applications for security. For example, Apple requires all apps listed on the Apple App Store to undergo rigorous reviews and security tests to ensure their security. Many companies also hire professional security testing teams or use third-party security testing companies to conduct security testing for mobile applications.

4. Conclusion

The research results of this paper mainly include the following aspects:

1.Deeply explored the principles and methods of software security testing technology, systematically summarized the various stages and test methods of software security testing, and provided the theoretical basis and guidelines for the practical application of software security testing.

2.Introduced a variety of tools and technologies for software security testing, including static analysis, dynamic analysis, fuzzy test and vulnerability scanning, etc., which provide technical support and implementation means for the practical application of software security testing.

3.Focus on the practical application of software security test, including how to develop test plan and test strategy, how to choose test tools and technology, how to test execution and test results analysis, etc., and combined with the case introduces the actual test process need to pay attention to the problems and skills, for the practice of software security test provides specific guidance.

4.The advantages and disadvantages of software security testing technology are summarized, and the direction that needs to be further improved and developed is indicated, which provides a reference for future research and practice.

In conclusion, the research results of this paper have certain value and significance in both theory and practice, and contribute to promoting the development and application of software security testing[4].

Acknowledgements

I would like to express my heartfelt gratitude to all those who have contributed to the completion of this research. My deepest appreciation goes to my supervisor for providing invaluable guidance, support, and encouragement throughout the entire research process. I also extend my thanks to all the interviewees who generously shared their time and insights with me. Additionally, I acknowledge the library staff for their assistance in sourcing relevant literature and resources. Finally, I would like to thank my family and friends for their unwavering support and encouragement during this challenging journey.

References

[1] Burgess S, Mason A M, Grant A J, et al.

Using genetic association data to guide drug discovery and development: Review of methods and applications[J]. The American Journal of Human Genetics, 2023, 110(2): 195-214.

[2] Cinar A C, Kara T B. The current state and future of mobile security in the light of the recent mobile security threat reports[J]. Multimedia Tools and Applications, 2023: 1-13.

[3] Liu W, Hua M, Deng Z, et al. A systematic survey of control techniques and applications: From autonomous vehicles to connected and automated vehicles[J]. ar**v preprint ar**v:2303.05665, 2023.

[4] Lin Y, Ma J, Wang Q, et al. Applications of machine learning techniques for enhancing nondestructive food quality and safety detection[J]. Critical Reviews in Food Science and Nutrition, 2023, 63(12): 1649-1669.

