Journal of Electronic Research and Application, 2024, Volume 8, Issue 6
http://ojs.bbwpublisher.com/index.php/JERA
ISSN Online: 2208-3510
ISSN Print: 2208-3502

Analysis of Software Development and Operation Measures from the Perspective of Security Technology

Yan Gao*

Tencent Americas, Los Angeles 90066, California, United States

Corresponding author: Yan Gao, gao.yan.gg1994@gmail.com

Abstract: Security technology is crucial in software development and operation in the digital age. Secure software can protect user privacy and data security, prevent hacker attacks and data breaches, ensure legitimate business operations, and protect core assets. However, the development process often faces threats such as injection attacks, Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF), mainly due to code vulnerabilities, configuration errors, and risks from third-party components. To meet these challenges, this paper discusses the application of security technology in development and operation, emphasizing security requirements analysis, design principles, coding practices, and testing during the development phase. Along with focusing on environmental configuration, continuous monitoring, emergency response, disaster recovery, and regular auditing and updating during the operation phase. These measures can significantly enhance the security of software systems and protect user and corporate data.

Keywords: Security technology; Software development; Operation and maintenance

Online publication: December 2, 2024

1. Introduction

With the rapid development of information technology, software systems have become an important support for corporate operations and user services. However, as software systems are widely applied, the security threats they face are becoming increasingly severe. Security incidents such as hacker attacks and data breaches occur frequently, causing significant losses to users and enterprises. Therefore, how to effectively integrate security technology into the software development and operation process to ensure the security of software systems has become a focal point of industry attention. This paper aims to explore the application of security technology in software development and operation, providing strong support for building secure and stable software systems.

2. The importance of software development security

Secure software holds immense significance in the digital era. It not only effectively protects user privacy and data security, preventing hacking attacks and data breaches, but also ensures that user's personal information and sensitive data are not illegally obtained or exploited, thereby gaining deep user trust. Following mandatory regulations such as the "Basic Requirements for Information System Security Protection Classification," software security development is a technical benchmark and a prerequisite for legal and compliant business operations [1]. Furthermore, secure development is crucial for protecting corporate core data assets, the foundation for business survival and development. Once leaked or illegally used, these data can lead to significant economic losses and reputational damage. Therefore, through secure development, businesses can ensure the security and integrity of their core data assets. Ultimately, secure software significantly enhances the user experience, strengthening users' trust in the software. This trust translates into the brand image and market competitiveness of the enterprise, giving it an advantageous position in the fiercely competitive market.

3. Major security threats faced by software development

3.1. Injection attacks: a threat that bypasses security controls

Injection attacks deceive applications into performing unauthorized actions by inputting malicious data. Structured query language (SQL) injection and command injection are typical methods of such attacks. They can bypass the security controls of applications, directly access or tamper with database data, and even execute system commands, posing a serious threat to system integrity and confidentiality [2].

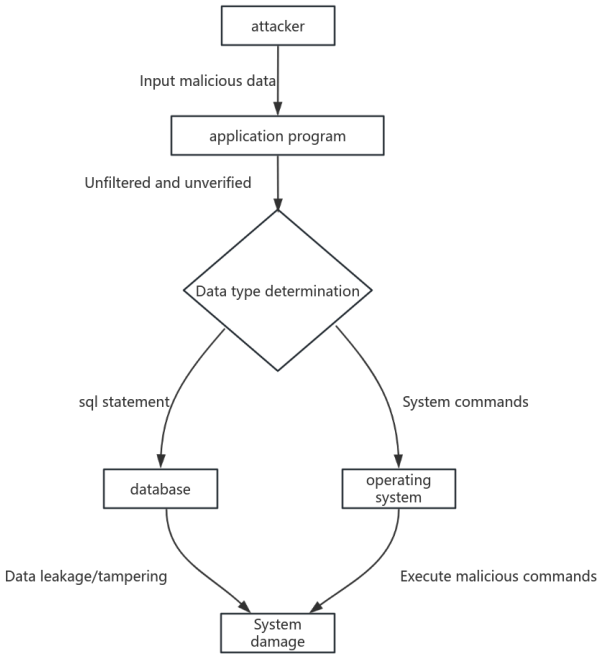

Figure 1. The basic flow of injection attacks

3.2. Cross-Site Scripting (XSS): a covert and harmful attack

Cross-Site Scripting attacks exploit vulnerabilities in applications that fail to adequately filter and encode

user inputs, injecting malicious script code into web pages. When other users browse these tampered pages, the malicious scripts execute, stealing sensitive user information, hijacking user sessions, or performing other malicious actions. XSS attacks are highly covert and harmful, making them difficult to detect and defend against promptly.

3.3. Cross-Site Request Forgery (CSRF): an attack exploiting user identity

Cross-Site Request Forgery is an attack method that exploits a user's currently logged-in identity to send malicious requests to the server on their behalf. By constructing specially crafted web pages or links, attackers deceive users into clicking on them, thereby executing unauthorized actions such as transfers or password changes without the user's knowledge. CSRF attacks directly threaten user data security and operational safety, causing significant losses to users [3].

3.4. Root cause analysis of security threats

The origins of these security threats are diverse but primarily include code vulnerabilities, incorrect security configurations, and insecure third-party components. Developers lacking security awareness or skills may result in undiscovered security vulnerabilities in the code. During the deployment and configuration of software systems, improper configurations or failure to follow best security practices can also lead to potential security risks. Furthermore, modern software development relies on a vast array of third-party components and libraries, which may themselves contain security vulnerabilities or have not undergone sufficient security testing. If developers do not conduct thorough security reviews, they may introduce potential security risks into their software systems.

4. Application of security technologies in software development

4.1. Security requirements analysis

Security requirements analysis is the first step in software development security, relying on in-depth user research. Through technical means such as questionnaires and in-depth interviews, developers can accurately capture users' specific expectations for security, which are then directly translated into security requirements for software design [4]. For example, online questionnaires can be conducted using tools like SurveyMonkey to widely collect user opinions. At the same time, competitive analysis, utilizing tools such as Gartner's Magic Quadrant and Open Worldwide Application Security Project (OWASP) Top Ten, systematically evaluates the security of similar products in the market, identifying competitors' security strengths and weaknesses to provide insights for the security design of one's software. Furthermore, developers need to thoroughly study software security standards and norms established by national and industry bodies, such as the International Organization for Standardization (ISO) 27001 and the Health Insurance Portability and Accountability Act (HIPAA), using these standards as benchmarks for security requirements identification to ensure that the software product meets relevant security requirements from the outset.

4.2. Security design principles

Security design principles occupy a central position in software development. The principle of least privilege, implemented through technologies such as Role-Based Access Control (RBAC), ensures that each user or system component is granted only the minimum permissions necessary to complete specific tasks. For

example, in database management, Oracle's permission management system can be used to allocate appropriate permissions based on user roles and needs, preventing permission abuse. Security isolation and layered design are achieved through technologies such as Docker containerization and microservices architecture, isolating system components or user data to prevent the spread of security issues. Layered design, such as the Model-View-Controller (MVC) architecture, enhances system security by dividing different layers and strictly regulating interface communication. Data encryption and transmission security rely on strong encryption algorithms like Advanced Encryption Standard (AES) and Rivest-Shamir-Adleman (RSA), as well as transmission security protocols such as Hypertext Transfer Protocol Secure (HTTPS), Secure Sockets Layer (SSL)/Transport Layer Security (TLS), to ensure data security during storage and transmission [5].

4.3. Secure coding practices

Secure coding practices are crucial for building secure applications. Input validation and output encoding are implemented through techniques such as regular expressions and prepared statements, rigorously validating and filtering all user inputs to prevent malicious data injection. For example, using Java's PreparedStatement class with parameterized queries avoids SQL injection. Output encoding, through techniques like HyperText Markup Language (HTML) escaping and Uniform Resource Locator (URL) encoding, prevents browsers from misinterpreting content as executable code, thereby avoiding XSS attacks. Developers should use coding libraries that have undergone rigorous security reviews, such as Apache Commons and OWASP's Enterprise Security API (ESAPI), to reduce security errors. Secure Application Programming Interface (API) design and usage ensure the security of API interfaces through authentication technologies like Open Authorization (OAuth) and JSON Web Token (JWT), as well as RESTful API design follows the principles of Representational State Transfer (REST). Simultaneously, protocols such as HTTPS and TLS are utilized to safeguard the security of data transmission and storage, ensuring that only legitimate users can access APIs and limiting access to resources within the scope of user permissions [6].

4.4. Security testing

Security testing serves as the last line of defense in ensuring the security of software systems. Static code analysis, through tools like SonarQube and FindBugs, conducts in-depth inspections of source code or binary code to identify potential security vulnerabilities, providing developers with detailed vulnerability reports and remediation suggestions. Dynamic testing, on the other hand, uses tools like Burp Suite and OWASP Zed Attack Proxy (ZAP) to simulate attacks or trigger security issues through abnormal inputs. Fuzzing testing is a typical method within this category, utilizing tools like Peach Fuzzer and American Fuzzy Lop to perform fuzz testing on software and uncover potential vulnerabilities. Penetration testing simulates hacker attacks to assess the overall security of the system, employing tools like Metasploit and Nmap for vulnerability scanning and attack simulation to discover and remediate security vulnerabilities. Designing comprehensive and targeted security test cases is key to ensuring the effectiveness of testing, leveraging test management tools like TestRail and JUnit to cover all security boundary conditions with repeatability and verifiability. During execution, combining automation with manual testing, meticulously recording results, and using defect tracking systems like Jira and Bugzilla facilitates timely communication with developers, ensuring prompt issue resolution and jointly enhancing the security of the software system.

5. Application of security technologies in software operations and maintenance

5.1. Environment security configuration

Environment security configuration is the cornerstone for building a secure and stable Information Technology (IT) environment, encompassing meticulous server and database security configurations, secure network architecture design, and the formulation and implementation of security policies and firewall rules. At the server security configuration level, it is necessary to close unnecessary service ports, set passwords using strong encryption algorithms, and change them regularly. Additionally, automated tools should be utilized to promptly install system patches to guard against known vulnerabilities. In terms of database security, strict access control policies should be implemented, assigning user permissions based on the principle of least privilege. Regular data backups should be performed, with secure storage facilities such as tape libraries or cloud storage used to ensure the reliability of backup data. In the design of secure network architectures, Virtual Local Area Network (VLAN) technology is employed to logically segment the network, reducing the risk of risk propagation. SSL/TLS protocols are used to encrypt remote access traffic, ensuring secure data transmission. Firewall rules are finely configured according to security policies and business needs, utilizing firewall devices like Cisco and Checkpoint to achieve dynamic adjustment of Access Control Lists (ACLs), ensuring effective implementation of security policies.

Table 1. Environment security configuration details

Environment security configuration checklist	Configuration item	Configuration standard
Server security configuration	Disable unnecessary service ports	Only keep necessary service ports open to reduce the potential attack surface
	Password settings	Use strong encryption algorithms (e.g., SHA-256) for passwords, set complex passwords, and change them regularly
	System patch management	Use automation tools (e.g. Ansible, Puppet) to promptly install system patches and mitigate known vulnerabilities
Database security configuration	Access control policy	Implement strict access control policies, allowing only authorized users to access
	User privilege assignment	Apply the principle of least privilege, assigning appropriate permissions based on user roles and needs
	Data backup and storage	Perform regular data backups and use secure storage facilities such as tape libraries or cloud storage to ensure the reliability of backup data
Network architecture security design	VLAN segmentation	Use VLAN technology to logically segment the network, reducing the risk of risk propagation
	Remote access encryption	Encrypt remote access traffic using SSL/TLS protocols to ensure data transmission security
	Firewall rule configuration	Fine-tune firewall rules based on security policies and business needs, using firewall devices like Cisco, Checkpoint, etc., to implement dynamic Access Control Lists (ACLs)

5.2. Continuous security monitoring

Continuous security monitoring serves as the defensive line for information system security, integrating log management and analysis, Intrusion Detection Systems (IDS)/Intrusion Protection Systems (IPS), and performance and security monitoring. In terms of log management, tools like Splunk and Elasticsearch,

Kibana, Beats, and Logstash (ELK Stack) are employed for centralized storage, indexing, and searching of logs, with stringent access controls and encryption measures in place to ensure log integrity and confidentiality. For IDS/IPS, open-source or commercial systems such as Snort and Suricata are deployed to monitor network traffic and system activities in real-time, using technologies like signature matching and behavioral analysis to identify and block potential threats. Performance monitoring tools like Nagios and Zabbix are used to monitor key indicators such as response time, throughput, Central Processing Unit (CPU), and memory utilization in real-time, while security monitoring leverages technologies like vulnerability scanners and File Integrity Monitoring (FIM) to detect security vulnerabilities and abnormal behaviors, triggering alerts and initiating automatic or manual remediation measures upon detection.

5.3. Emergency response and disaster recovery

Emergency response and disaster recovery are crucial for ensuring business continuity, encompassing event reporting, analysis, handling, recovery, and subsequent improvement. Event reporting is automated through Security Information and Event Management (SIEM) systems, ensuring that all security events are promptly captured and reported. During event analysis, resources such as threat intelligence and vulnerability databases are utilized to thoroughly analyze the nature and scope of the event, employing technologies like Distributed Denial-of-Service (DDoS) protection and malware analysis to quickly locate and eliminate threats. In the recovery stage, techniques such as backup restoration and failover are used to ensure that systems and data are swiftly restored to normal status. The subsequent improvement stage involves regular reviews, revisions of security policies, and other measures to continuously optimize the emergency response process. In terms of data backup and recovery strategies, multiple backup solutions like tape backup and cloud backup are employed to ensure rapid data recovery in case of loss or damage. Failover and disaster recovery mechanisms are implemented through measures such as deploying redundant devices and establishing offsite disaster recovery centers, ensuring seamless business continuation to backup systems in the event of primary system failures, thereby minimizing losses.

5.4. Regular security audits and updates

Regular security audits and updates are a routine aspect of software system security, encompassing audits of security policies and compliance, software and dependency library updates, and security training and awareness enhancement. For security policy and compliance audits, automated audit tools like Nessus and OpenVAS are used to periodically review access control policies, password policies, data encryption policies, and more, ensuring compliance with regulations and standards such as HIPAA and General Data Protection Regulation (GDPR). In terms of software and dependency library updates, automated update tools like Ansible and Puppet are employed to ensure timely updates of operating systems, databases, applications, and third-party libraries and frameworks, with sandbox testing, regression testing, and other measures conducted before updates to ensure stability. For security training and awareness enhancement, regular security lectures, seminars, and simulated attack exercises (like Capture The Flag (CTF) competitions) are held to enhance the operations team's knowledge of the latest threats and defense skills, fostering a sense of security responsibility and sensitivity, and jointly maintaining the secure and stable operation of software systems.

6. Conclusion

In summary, security technologies play a pivotal role in software development and operations. By implementing security measures at various stages of development and operations, such as requirements analysis, design principles, coding practices, testing, environment configuration, continuous monitoring, emergency response, disaster recovery, and regular audits, the security of software systems can be significantly enhanced, protecting user and enterprise data. Facing constantly evolving security threats and technological advancements, we must keep up with new developments in security technologies, continuously optimizing and improving our security protection systems to address the severe challenges.

Disclosure statement

The author declares no conflict of interest.

References

[1] Werder K, Li Y, Maedche A, et al., 2021, Software Development Process Ambidexterity and Project Performance: A Coordination Cost-Effectiveness View. IEEE Transactions on Software Engineering, 47(4): 836–849.

[2] Liyan S, Leandro L, Minku A, 2023, Procedure to Continuously Evaluate Predictive Performance of Just-In-Time Software Defect Prediction Models During Software Development. IEEE Transactions on Software Engineering, 49(2): 646–666.

[3] Elvan K, Eric G, Arie VD, et al., 2022, Factors Affecting On-Time Delivery in Large-Scale Agile Software Development. IEEE Transactions on Software Engineering, 48(9): 3573–3592.

[4] Bernsmed K, Cruzes DS, Jaatun MG, et al., 2022, Adopting Threat Modeling in Agile Software Development Projects. The Journal of Systems and Software, 2022: 183.

[5] Patent Issued for Software Development Environment with Compilation and Read-Evaluate-Print-Loop Operations (USPTO 11256481), 2022: 5338–5342.

[6] Khan SU, Khan AW, Khan F, et al., 2022, Critical Success Factors of Component-Based Software Outsourcing Development From Vendors' Perspective: A Systematic Literature Review. IEEE Access, 10: 1650–1658.