THE ROLE AND IMPORTANCE OF SOFTWARE IN INFORMATION SECURITY

Aliev Avazbek Ulugbek Ugli
Teacher at the Digital Technologies and Information Security of the Academy of the Ministry of Internal Affairs of the Republic of Uzbekistan.
Dzhamatov Mustafa Khatamovich
Teacher at the Digital Technologies and Information Security of the Academy of the Ministry of Internal Affairs of the Republic of Uzbekistan

https://doi.org/10.5281/zenodo.18037737

Annotation. *With the rapid development of information technology, software security has become a key component of modern information security. Today, more than 90% of cyberattacks are carried out through software vulnerabilities. This article analyzes the role of software in information security, its importance and future trends.*

Key words: *Password-based, network, virus, Creeper, ISO, Security, SSDLC.*

HISTORICAL EVOLUTION OF SOFTWARE SECURITY

Early Period (1960-1980)

The concept of software security first emerged in the 1960s. During this period, the main focus was on:

- Physical security: Focus on physical security of server rooms.
- Simple authentication: Password-based systems.
- Limited network connectivity: Local area networks, limited use of the internet.

Table 1: Initial security measures

Year	Incident	Result
1971	The first computer virus "Creeper"	The emergence of antivirus programs
1973	Development of the TCP/IP protocol	Emergence of network security problems
1976	Diffie-Hellman key exchange algorithm	The development of cryptography

Internet era (1990-2000)

With the widespread use of the Internet, software security has reached a new level:

- Web security: emergence of attacks such as SQL injection, XSS
- Antivirus software: became widely used.
- Security standards: ISO 17799, Common Criteria appeared.

Modern era (2000-2020)

During this period, software security has become strategically important:

- SDLC (Software Development Life Cycle): Integrating security from the earliest stages;
- DevSecOps: Integration of Development, Security, Operations;
- Cloud Security: Security of cloud technologies.

Current era (2020-present)

• AI-powered security: Analytics using artificial intelligence;
• Zero Trust: "Never trust, always verify" principle;
• Supply Chain Security: Supply chain security.

SOFTWARE VULNERABILITIES AND THEIR CLASSIFICATION

OWASP Top 10 - 2023

Table 2: OWASP Top 10 2023

Place	Type of weakness	Impact	Distribution
1	Broken Access Control	High	Wide
2	Cryptographic Failures	High	Average
3	Injection	High	Wide
4	Insecure Design	High	Average
5	Security Misconfiguration	Average	Very wide
6	Vulnerable Components	Average	Wide
7	Identification Failures	Average	Wide
8	Software Integrity Failures	Past	Limited
9	Security Logging Failures	Average	Wide
10	Server-Side Request Forgery	Average	Growing

Types of software vulnerabilities and code-level vulnerabilities

• **Buffer Overflow**: Writing more than the specified amount of data to memory;
• **Memory Leaks:** Failure to release memory properly;
• **Race Conditions:** Time synchronization problems.

Design-level vulnerabilities

• **Authentication Bypass:** Bypass authentication
• **Privilege Escalation:** Increase privileges
• **Business Logic Errors:** Business logic errors

Configuration vulnerabilities

• **Default Passwords:** Default passwords
• **Unnecessary Services:** Running unnecessary services
• **Insecure Permissions:** Insecure permissions

Statistical data according to 2023 research

• There are an average of 15 security vulnerabilities in every software package
• 70% of vulnerabilities can be prevented during software development
• Only 30% of companies conduct regular security testing
• Average cost of fixing security vulnerabilities: $5,000-50,000

SECURE SOFTWARE DEVELOPMENT METHODOLOGIES

Secure Software Development Life Cycle (SSDLC)

SSDLC - a methodology for integrating security into all phases of software development.

Phase 1: Requirements Definition

• **Security Requirements:** Defining Security Requirements
• **Threat Modeling:** Threat Modeling
• **Risk Assessment:** Risk Assessment

Phase 2: Design

• **Secure Architecture:** Designing a secure architecture
• **Security Patterns:** Applying security patterns
• **Design Review:** Reviewing the design from a security perspective

Phase 3: Development

• **Secure Coding Standards:** Secure coding standards
• **Static Analysis:** Static analysis
• **Code Review:** Code review

Phase 4: Testing

• **Dynamic Analysis**
• **Penetration Testing**
• **Vulnerability Scanning**

Phase 5: Production

• **Secure Deployment:** Secure Deployment
• **Configuration Management:** Configuration Management
• **Monitoring:** Monitoring

Phase 6: Maintain and Update

• **Patch Management**
• **Incident Response**
• **Continuous Improvement**

DevSecOps-Integration of Development, Security, and Operations.

Table 3: DevSecOps Principles

Principle	Description	Benefit
Shift Left	Moving security to early stages	Reduce costs by 100 times
Automation	Automate security processes	Speed increase
Collaboration	Collaboration between team members	Better risk management
Continuous Security	Continuous security monitoring	Real-time risk identification

Security integration in CI/CD

The following security tools are integrated into modern CI/CD pipelines:

1. SAST (Static Application Security Testing):

o Static analysis of code

o Example: SonarQube, Checkmarx

2. DAST (Dynamic Application Security Testing):

o Dynamic analysis of a working application

o Example: OWASP ZAP, Burp Suite

3. SCA (Software Composition Analysis):

o Analysis of third-party libraries

o Example: Snyk, WhiteSource

4. Container Security:

o Verification of container security

o Example: Anchore, Clair

5. Infrastructure as Code Security:

o Verification of IaC files

o Example: Terrascan, Checkov

SOFTWARE SECURITY STANDARDS AND FRAMEWORKS

INTERNATIONAL STANDARDS

ISO/IEC 27034

• Software Security Management

• Application Security Controls Framework

NIST SP 800-53

• Security and privacy controls

• Risk management

OWASP ASVS (Application Security Verification Standard)

• Software Security Verification Standard

• 3-level verification

Industrial frames

BSIMM (Building Security In Maturity Model)

• 119 practices

• 12 domains

• 4 levels of complexity

SAMM (Software Assurance Maturity Model)

• 15 practices

• 4 management domains

• 3 levels of practice

CONCLUSION

Software security has become an integral part of modern information security, and 70% of vulnerabilities can be prevented during software development, and DevSecOps methodology significantly increases efficiency, AI and automation define the future of software security. Software developers need to master secure coding practices, regular training, and code review for team code review.

REFERENCES

1. Howard, M., & LeBlanc, D. (2003). Writing Secure Code. Microsoft Press.
2. McGraw, G. (2006). Software Security: Building Security In. Addison-Wesley.
3. OWASP Foundation. (2023). OWASP Top Ten 2023.
4. NIST. (2022). Secure Software Development Framework (SSDF).
5. ISO/IEC 27034-1:2011. Information technology - Security techniques.
6. "Raqamli O'zbekiston-2030" strategiyasi.
7. Axborot xavfsizligi milliy standartlari (O'zDSt 2847-2850).
8. Абдазимов С. Особенности применения технологии блокчейн в сфере информационной безопасности и финансовых операций //Modern Science and Research. – 2025. – Т. 4. – №. 5. – С. 1480-1482.
9. Джаматов М., Абдазимов С. Основные принципы создания программного обеспечения для систем распознавания объектов //Modern Science and Research. – 2025. – Т. 4. – №. 1. – С. 32-38.
10. Мирзаева М. Б., Абдазимов С. З. Использование технологий искусственного интеллекта в сфере высшего образования. – 2022.