Program Security

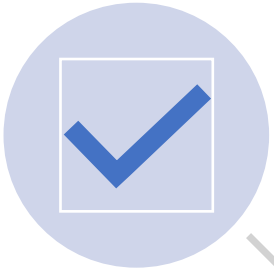
By

Prof. Muhammad Iqbal Bhat

Government Degree College
Beerwah

Topics

DEFINITION OF PROGRAM SECURITY



GOALS OF PROGRAM SECURITY

EXAMPLES OF PROGRAM SECURITY BREACHES

OVERVIEW OF SECURE PROGRAMMING PRACTICES

Recent Attacks:

Incident	Description	Date	Impact
SolarWinds supply chain attack	Hackers compromised SolarWinds' software supply chain to distribute malware that gave them access to hundreds of organizations' networks, including government agencies and Fortune 500 companies.	December 2020	Impact still being assessed, but potentially significant compromise of confidential data, potential for espionage, and reputational damage for affected organizations.
Colonial Pipeline ransomware attack	A ransomware attack shut down Colonial Pipeline's computer systems, causing disruptions in the fuel supply chain on the East Coast of the United States.	May 2021	Colonial Pipeline paid $4.4 million in ransom to the attackers. Short-term fuel shortages, long-term damage to the company's reputation, and potential financial losses.
Microsoft Exchange Server vulnerabilities	Several zero-day vulnerabilities were discovered in Microsoft Exchange Server, which allowed hackers to steal sensitive data from organizations' email servers.	March 2021	Impact still being assessed, but potentially significant compromise of confidential data, potential for espionage, and reputational damage for affected organizations.
Twitter hack	Hackers gained access to Twitter's internal systems and hijacked high-profile accounts, including those of Barack Obama and Elon Musk, to promote a bitcoin scam.	July 2020	Damage to the reputation of affected individuals and potential financial losses for those who fell victim to the scam.
Equifax data breach	Hackers accessed Equifax's database and stole personal and financial information of 143 million consumers.	May-July 2017	Equifax agreed to pay $700 million to settle claims related to the breach. Significant financial losses and reputational damage for Equifax and potential financial losses for affected consumers.

Program Security:

Program security is a set of practices, processes, and technologies used to protect computer programs from unauthorized access, modification, or destruction.

Programs can include any software, from standalone applications to complex systems and networks.

Program security is a subset of information security, which refers to the protection of all types of information assets, including data, software, hardware, and people.

Program security is a multifaceted discipline that encompasses a range of technical and non-technical areas, including cryptography, access control, risk management, incident response, and security awareness training.

Program security is essential in today's digital world due to the growing number and sophistication of cyber threats, which can lead to financial loss, reputation damage, and legal liability for organizations.

Program security involves both proactive measures, such as implementing security controls, and reactive measures, such as detecting and responding to security incidents.

Program security is often integrated into the software development lifecycle, with security requirements and testing being incorporated throughout the process.

Program security is not a one-time effort but an ongoing process, as new threats and vulnerabilities emerge and existing ones evolve over time.

Program security requires a holistic approach that considers not only technical factors but also organizational culture, policies, and governance structures.

Goals of Program Security:

Confidentiality: ensuring that information is kept confidential and is not disclosed to unauthorized parties.

Integrity: ensuring that information is accurate and has not been tampered with.

Availability: ensuring that information and services are available when needed.

Non-repudiation: ensuring that the actions of a user or system cannot be denied or disputed.

Authentication: verifying the identity of a user or system.

Authorization: granting or denying access to resources based on a user's identity and permissions.

Accountability: ensuring that users are held responsible for their actions.

Secure programming practices

Secure programming practices are techniques used to develop software that is resistant to security threats and vulnerabilities.

Secure programming practices should be applied throughout the software development lifecycle, including design, coding, testing, and deployment.

Secure programming practices are intended to prevent or mitigate various types of security vulnerabilities, such as buffer overflows, SQL injection, cross-site scripting (XSS), and race conditions.

Some common secure programming practices include input validation, output encoding, proper error handling, authentication and access control, encryption and decryption, and secure coding standards.

Input validation involves checking and sanitizing user input to ensure it meets certain criteria, such as type, length, and format.

Output encoding involves encoding output to prevent injection attacks, such as XSS attacks, which exploit vulnerabilities in web applications that allow user input to be included in web pages without proper encoding.

Continue…

Proper error handling involves handling errors in a way that does not expose sensitive information or provide information that could be used to exploit the system.

Authentication and access control involves verifying the identity of users and restricting access to resources based on their permissions and privileges.

Encryption and decryption involve protecting data by converting it into a code that cannot be easily read or understood without a decryption key.

Secure coding standards are guidelines or best practices for writing secure code that are often enforced through code reviews or automated tools.

Secure programming practices are constantly evolving as new threats and vulnerabilities emerge, so it is important for developers to stay up-to-date on the latest techniques and tools for secure programming.

Secure programming practices are increasingly important in the age of cloud computing, mobile computing, and Internet of Things (IoT) devices, where software systems are more interconnected and exposed to a wider range of security threats and vulnerabilities.

Threat Modeling:

Threat modeling is a process used to identify and prioritize potential threats and vulnerabilities in a software system or application.

The goal of threat modeling is to provide a structured approach to security risk assessment and help developers and security analysts identify and address security issues before they are exploited by attackers.

Threat modeling can be done at various stages of the software development lifecycle, from design to deployment, and can be applied to both new and existing software systems.

Threat modeling can also be automated using tools such as Microsoft Threat Modeling Tool, IriusRisk, and ThreatModeler.

Threat modeling is an important part of a comprehensive security program, but it is not a substitute for other security measures, such as access controls, encryption, and monitoring.

Threat Modeling Process:

Identify the system components and boundaries: This involves identifying the various components of the system and defining the boundaries of the system, including inputs and outputs.

Create a data flow diagram: This involves mapping out the flow of data through the system, including how data enters, moves through, and exits the system.

Identify threats and vulnerabilities: This involves identifying potential threats and vulnerabilities at each stage of the data flow, including input validation, authentication and access control, encryption, and error handling.

Prioritize and evaluate threats: This involves prioritizing the identified threats based on the potential impact and likelihood of occurrence and evaluating the effectiveness of existing security controls in mitigating these threats.

Mitigate the most critical threats: This involves implementing appropriate security measures to address the most critical threats identified during the threat modeling process.

Security Testing:

Security testing is a process used to evaluate the security posture of a software system or application by simulating various types of attacks and vulnerabilities.

The goal of security testing is to identify and address security vulnerabilities before they are exploited by attackers and to ensure that the software system or application meets the security requirements and standards.

Security testing can be done at various stages of the software development lifecycle, from design to deployment, and can be applied to both new and existing software systems.

Security testing can be done using various techniques and tools, such as penetration testing, vulnerability scanning, code review, and fuzz testing.

Penetration testing involves simulating attacks on a software system or application to identify vulnerabilities and assess the effectiveness of existing security controls.

Vulnerability scanning involves using automated tools to scan a software system or application for known vulnerabilities and misconfigurations.

Code review involves reviewing the source code of a software system or application to identify potential security vulnerabilities and weaknesses

Security Standards:

Security standards and guidelines are sets of rules, requirements, and best practices that define the minimum security requirements for a software system or application.

Security standards and guidelines are developed by various organizations and bodies, such as government agencies, industry associations, and standards organizations.

Security standards and guidelines provide a framework for ensuring the confidentiality, integrity, and availability of information and data within a software system or application.

Security standards and guidelines can cover various aspects of security, such as access controls, cryptography, network security, and software security.

Compliance with security standards and guidelines is often required by law or regulation, or by contractual obligations with customers or partners.

Adherence to security standards and guidelines can help organizations avoid security incidents and data breaches, reduce legal and financial risks, and improve customer trust and confidence.

Security standards and guidelines should be integrated into the software development lifecycle to ensure that security is built into the software system or application from the beginning.

Examples of Security Standards:

ISO/IEC 27001: This is an international standard that defines the requirements for an information security management system (ISMS).

NIST Cybersecurity Framework: This is a framework developed by the National Institute of Standards and Technology (NIST) that provides guidelines for improving cybersecurity risk management.

PCI DSS: This is a set of security standards developed by the Payment Card Industry Security Standards Council to ensure the security of credit card transactions.

OWASP Top Ten: This is a list of the top ten most critical web application security risks developed by the Open Web Application Security Project (OWASP).