Birla Institute of Technology & Science, Pilani
Work Integrated Learning Programmes Division
Second Semester 2020-2021

Mid-Semester Test
(EC-2 Regular)

Course No.	: SE ZG566
Course Title	: Secure Software Engineering
Nature of Exam	: Open Book
Weightage	: 30%
Duration	: 2 Hours
Date of Exam	: 11/03/2022 (FN)

No. of Pages = 1
No. of Questions = 5

Note to Students:
1. Please follow all the *Instructions to Candidates* given on the cover page of the answer book.
2. All parts of a question should be answered consecutively. Each answer should start from a fresh page.
3. Assumptions made if any, should be stated clearly at the beginning of your answer.

Q.1. Provide brief response (in 50 words) [2X6 = 12 Marks]

a. How does CIA triad apply to a telecom service provider?

The CIA triad is a widely recognized model used to describe the three fundamental pillars of information security: confidentiality, integrity, and availability. These three components are essential for ensuring the security of information and information systems.

In the context of a telecom service provider, the CIA triad can be applied as follows:

1. Confidentiality: Telecom service providers handle sensitive and confidential information such as customer data, financial data, and network configurations. Maintaining confidentiality is critical to protect this information from unauthorized access, theft or misuse. The telecom service provider must implement strong access controls, encryption, and secure communication channels to protect confidential information.

Telecom providers handle a lot of PII /sensitive data as part of KYC process for SIM purchase, location, financial data for billing and as part of consumer's calls - some confidential call history etc may be maintained. So they need measures to secure data during transit as well as data(customer records) at rest using end-to-end encryption, features like disabling location sharing etc.

Avoid cross connections.

2. Integrity: Telecom service providers must ensure the integrity of the information they handle. This means that the information must be accurate, complete, and trustworthy. Any unauthorized modification or alteration of information can have serious consequences, including loss of customer trust and credibility. The service provider must implement measures such as data validation, error detection, and access control to ensure the integrity of the information.

For sms integrity should be maintained i.e. no malicious actor should be able to intercept and change a message such as a message containing OTP.

Authentication using a session key should be used to establish a link.

Reordering of messages should be avoided.

3. Availability: Telecom service providers must ensure that their services and systems are available to customers at all times. Any disruption in service can have serious consequences, including loss of revenue, customer trust, and reputation. The service provider must implement measures such as redundancy, fault tolerance, and disaster recovery planning to ensure availability of services.

DDos attack like a flood of sms should be mitigated.
Congestion control should be used to keep the network available.
Telecom operators should abide by the SLA defined by TRAI or ITUT.

b. How does taint analysis help uncover vulnerabilities?

Taint analysis is a technique used in software security testing to identify vulnerabilities in code that could be exploited by attackers. Taint analysis works by tracking the flow of data through a program and identifying any inputs that can be manipulated by an attacker. By tracing the flow of these inputs, taint analysis can identify any points in the program where the inputs are used in ways that could lead to security vulnerabilities. Taint analysis helps uncover vulnerabilities in several ways:

1. Identifying input validation vulnerabilities: Taint analysis can identify inputs that are not properly validated by the program, leaving them vulnerable to injection attacks such as SQL injection or command injection. By tracing the flow of these inputs, taint analysis can identify any points in the program where the inputs are not properly sanitized or validated.
2. Identifying data leakage vulnerabilities: Taint analysis can identify points in the program where sensitive data is leaked or exposed to potential attackers. By tracing the flow of data, taint analysis can identify any points where sensitive data is transmitted over insecure channels or stored in unsecured locations.
3. Identifying authorization vulnerabilities: Taint analysis can identify points in the program where authorization checks are not properly enforced, allowing attackers to access resources or functionality that they should not have access to. By tracing the flow of data and identifying any inputs that are used in authorization checks, taint analysis can identify any points where these checks can be bypassed or circumvented.
4. Identifying security control vulnerabilities: Taint analysis can identify points in the program where security controls such as encryption, authentication, or access control are not properly implemented or enforced. By tracing the flow of data and identifying any inputs that are used in these controls, taint analysis can identify any points where these controls can be bypassed or circumvented.

In summary, taint analysis helps uncover vulnerabilities by identifying points in a program where input validation, data leakage, authorization, or security controls are not properly implemented or enforced. By tracing the flow of data through a program and identifying any inputs that can be manipulated by an attacker, taint analysis can help identify potential security vulnerabilities that could be exploited by attackers.

c. What is a malware? What is meant by zombie in context of malware?

A malware, short for malicious software, is a type of software that is specifically designed to damage, disrupt, or gain unauthorized access to a computer system or network. Malware can take many forms, including viruses, worms, trojans,

ransomware, and spyware. Malware can be used by cybercriminals to steal sensitive data, gain unauthorized access to systems, or disrupt operations.
A zombie, in the context of malware, is a computer or device that has been infected with malware and is under the control of a remote attacker. Zombies are also referred to as bots, botnets, or drones. Attackers can use zombies to perform a wide range of malicious activities, such as launching distributed denial of service (DDoS) attacks, sending spam emails, stealing sensitive data, or spreading malware to other systems.
Zombies are typically created by infecting a large number of computers or devices with malware, which then allows the attacker to control them remotely. Once a computer or device is infected, the malware establishes a connection to a command and control (C&C) server operated by the attacker. The attacker can then use the C&C server to issue commands to the infected devices, which can include launching attacks or stealing data.
Zombies are a significant threat to computer systems and networks because they can be used to launch large-scale attacks that are difficult to detect and mitigate. Attackers can use zombies to launch DDoS attacks that can overwhelm a network or server, making it unavailable to legitimate users. They can also use zombies to spread malware to other systems or steal sensitive data, such as passwords or financial information.
In summary, a malware is a type of software that is designed to damage, disrupt, or gain unauthorized access to a computer system or network. A zombie, in the context of malware, is a computer or device that has been infected with malware and is under the control of a remote attacker. Zombies are used by attackers to perform a variety of malicious activities, including launching DDoS attacks, sending spam emails, stealing sensitive data, and spreading malware to other systems.

d. Why is software security considered an asymmetric problem?

Software security is considered an asymmetric problem because it is much easier to find and exploit vulnerabilities in software than it is to identify and fix them. This is due to several factors:

1. Attackers have the advantage: Attackers only need to find one vulnerability in a piece of software to exploit it, while defenders must identify and fix all potential vulnerabilities. Attackers have the advantage of time and resources, and can use automated tools to scan for vulnerabilities at scale.
2. Complexity of software: Modern software systems are complex and often composed of many different components, making it difficult to identify all potential vulnerabilities. As software systems grow and become more interconnected, the number of potential vulnerabilities increases exponentially.
3. Limited resources for security: Organizations often have limited resources for software security, and may not have the expertise or budget to identify and fix all potential vulnerabilities. This can lead to a prioritization of security efforts, with some vulnerabilities being left unaddressed.
4. Human error: Many vulnerabilities are introduced into software systems through human error, such as coding mistakes or misconfigurations. As humans are fallible, it is difficult to eliminate all potential vulnerabilities through manual code review and testing.
5. Rapid pace of software development: Software development is often done at a rapid pace, with new features and functionality being added quickly. This can lead to security being an afterthought, with vulnerabilities only being discovered after the software has been released.

In summary, software security is considered an asymmetric problem because attackers have the advantage in identifying and exploiting vulnerabilities, software

systems are complex and difficult to secure, and organizations have limited resources for security. This makes it a challenging problem to solve, and requires a multi-faceted approach that includes secure coding practices, automated testing, and ongoing monitoring and maintenance.

refer to page 28-30 in merged doc

e. Differentiate interpretation of non-repudiation in contexts of security and privacy.

Non-repudiation is a concept used in both security and privacy contexts, but with different interpretations.
In the context of security, non-repudiation refers to the ability to prove that a particular action was taken by a specific user, and that the user cannot deny having taken that action. Non-repudiation is often used in the context of digital signatures and electronic transactions, where it is important to ensure that the parties involved cannot deny having signed a document or made a transaction. In this context, non-repudiation is a security measure that helps prevent fraud and disputes.
In the context of privacy, non-repudiation refers to the ability of a user to deny having taken a particular action, and to prevent others from proving otherwise.
Non-repudiation is often used in the context of anonymous communication, where users may not want to reveal their identity or actions to others. In this context, non-repudiation is a privacy measure that helps protect users from being identified or tracked.
To summarize, non-repudiation has different interpretations depending on the context. In the context of security, non-repudiation is used to prove that a particular action was taken by a specific user, while in the context of privacy, non-repudiation is used to enable users to deny taking a particular action and protect their identity.

f. Differentiate an entry in CVE from CWE.

CVE (Common Vulnerabilities and Exposures) and CWE (Common Weakness Enumeration) are both standards used in software security to identify and categorize vulnerabilities and weaknesses in software systems.
The main difference between CVE and CWE is the level of abstraction they operate at. CVE is a dictionary of standardized names for vulnerabilities and exposures in software systems. Each CVE entry corresponds to a specific vulnerability or exposure that has been identified and assigned a unique identifier. CVE provides a standard way for software vendors, researchers, and security practitioners to reference and discuss specific vulnerabilities and exposures.
CWE, on the other hand, is a taxonomy of common software weaknesses and vulnerabilities. CWE provides a standard way to categorize and describe different types of weaknesses and vulnerabilities in software systems. Each CWE entry corresponds to a specific type of weakness or vulnerability, such as buffer overflow, SQL injection, or cross-site scripting.
In summary, CVE provides a standardized way to reference and discuss specific vulnerabilities and exposures, while CWE provides a standardized way to categorize and describe different types of weaknesses and vulnerabilities in software systems. CVE entries are specific to a particular vulnerability or exposure, while CWE entries are broader and cover multiple vulnerabilities and weaknesses that share similar characteristics.

Q.2. What are Secure Implementation Patterns? How do they enhance software quality? Explain with examples (in 100 words). [4 Marks]

Secure Implementation Patterns are design patterns that are specifically designed to enhance the security of software systems. These patterns provide guidance and best practices for implementing secure software and can help mitigate common security vulnerabilities.
Secure Implementation Patterns enhance software quality by reducing the likelihood of security vulnerabilities and improving the overall quality of the software. By following established security patterns, developers can ensure that their software is built with security in mind from the very beginning. This can help prevent security vulnerabilities and reduce the risk of data breaches or other security incidents.
For example, the Input Validation pattern is a Secure Implementation Pattern that ensures that all input to a software system is properly validated and sanitized. This helps prevent common vulnerabilities such as SQL injection and cross-site scripting.
Another example is the Access Control pattern, which provides guidelines for implementing secure access controls to ensure that only authorized users have access to sensitive data or functionality.
In summary, Secure Implementation Patterns provide guidance and best practices for implementing secure software and can help enhance software quality by reducing the likelihood of security vulnerabilities and improving overall security. By following established patterns, developers can build software that is more secure, reliable, and resilient.

Q.3. How does CMU SQUARE process differ from OWASP SAMM and Microsoft SDL w.r.t. Requirements Engineering? Explain (in 100 words). [4 Marks]

CGPT

CMU SQUARE, OWASP SAMM, and Microsoft SDL are all software security frameworks that provide guidance and best practices for implementing secure software systems. However, they differ in their approach to Requirements Engineering.
CMU SQUARE places a strong emphasis on Requirements Engineering, with a dedicated step in the process for defining and analyzing requirements.
The Requirements Engineering step includes techniques such as use case analysis, scenario-based analysis, and risk-based analysis to identify and prioritize security requirements.
OWASP SAMM also includes Requirements Engineering as a key element of the framework, but it focuses on identifying security-related business objectives and mapping them to security requirements. The framework provides guidance for conducting a security requirements workshop and defining security requirements based on the organization's specific needs.
Microsoft SDL, on the other hand, does not have a dedicated step for Requirements Engineering, but instead incorporates security requirements into each phase of the software development lifecycle. The framework includes guidance and checklists for each phase of the development process to ensure that security requirements are considered and implemented.
In summary, CMU SQUARE, OWASP SAMM, and Microsoft SDL all address Requirements Engineering in their software security frameworks, but they differ in their approach. CMU SQUARE places a strong emphasis on Requirements Engineering, OWASP SAMM focuses on identifying security-related business objectives, and Microsoft SDL incorporates security requirements into each phase of the development process.

SAGE

CMU SQUARE, OWASP SAMM, and Microsoft SDL are all software security frameworks that aim to improve the security of software development processes. However, they differ in their approaches to requirements engineering.
CMU SQUARE places a strong emphasis on gathering and analyzing requirements in order to identify potential security risks and vulnerabilities. The process involves identifying stakeholders, gathering requirements, and analyzing potential risks and vulnerabilities in each requirement.
OWASP SAMM also emphasizes the importance of requirements engineering, but focuses more on aligning security objectives with business objectives. It involves identifying security requirements at different levels of the software development lifecycle and aligning them with business objectives.
Microsoft SDL, on the other hand, focuses on integrating security requirements into the software development process. This involves identifying security requirements and integrating them into the software development lifecycle, including design, coding, testing, and release.
Overall, while all three frameworks prioritize security requirements, they differ in their approaches to requirements engineering, with CMU SQUARE placing a greater emphasis on risk analysis and OWASP SAMM focusing on alignment with business objectives. Microsoft SDL emphasizes integrating security requirements into the software development process.

Q.4. You are security consultant for a courier company. They like to automate their operations. Perform Threat Modelling and Build necessary artifacts, and propose recommendations for their development team. [5 Marks]

CGPT

Performing a threat modeling exercise for the courier company's automation project can help identify potential security risks and vulnerabilities early in the development process. The following is a high-level overview of the threat modeling process and the artifacts that should be produced:

1. Identify the system components and boundaries: Identify the different components of the automation system, such as the web application, mobile application, backend servers, and databases. Define the boundaries between these components and any external systems that interact with them.
2. Create a data flow diagram: Create a diagram that shows how data flows through the system components, including inputs, outputs, and storage. This diagram will help identify potential attack surfaces and points of vulnerability.
3. Identify threats: Identify potential threats to the system, such as unauthorized access, injection attacks, denial of service, and data theft. Consider both technical threats (e.g., software vulnerabilities) and non-technical threats (e.g., social engineering).
4. Assess risks: Assess the likelihood and impact of each identified threat. Prioritize the threats based on their potential impact and likelihood.
5. Develop mitigation strategies: Develop mitigation strategies to address each identified threat. These strategies may include technical controls (e.g., encryption, access controls), process controls (e.g., training, monitoring), or a combination of both.

6. Produce artifacts: Produce a report that summarizes the results of the threat modeling exercise, including the identified threats, assessed risks, and mitigation strategies. Provide recommendations to the development team for implementing the mitigation strategies and improving the security of the system.

Recommendations for the development team may include:

1. Implementing secure coding practices, such as input validation and output encoding, to prevent injection attacks.
2. Using secure communication protocols, such as HTTPS, to protect data in transit.
3. Implementing access controls to limit access to sensitive data and functionality.
4. Conducting regular vulnerability assessments and penetration testing to identify and address security vulnerabilities.
5. Providing security training to development team members to ensure they are aware of security best practices and potential threats.

By performing a threat modeling exercise and implementing these recommendations, the courier company can enhance the security of their automation system and reduce the risk of security incidents.

Claude

Here are the steps I would take as a security consultant:

1. Conduct threat modeling workshop with development team:
   - Diagram key system components and data flows (data flow diagrams)
   - Identify threats for each component using STRIDE analysis
   - Note potential impacts and security requirements
   - Document findings and diagrams as threat modeling artifacts
2. Review system requirements and design:
   - Verify that security requirements from threat modeling are addressed
   - Identify and fill any gaps in security controls
   - Recommend design changes if needed to mitigate threats
   - Document security recommendations and get team's agreement
3. Review and test security controls:
   - Assess effectiveness of controls in requirements and design
   - Recommend improvements or additional controls if needed
   - Perform penetration testing and security audits to uncover vulnerabilities
   - Work with team to fix issues and ensure controls are working as intended
4. Monitor security:
   - Continue assessing risks from evolving threats and system changes
   - Provide guidance on risk management and control enhancements
   - Perform periodic security reviews and testing to sustain security

This would help the courier company develop a more secure automated system and process. Let me know if you would like me to elaborate on any of the steps.

Q.5. You are a Security Professional. Your client, who is building an Information System, needs to choose between penetration and fuzz testing (choose only one). Prepare a questionnaire (with at least 5 questions) to the client to enable you to make recommendation. Justify the questionnaire. [5 Marks]

CGPT

Questionnaire:

1. What is the primary objective of your information system?
2. What is the expected level of security required for your information system?
3. How critical is your information system to your business operations?
4. What is your current level of security expertise and resources?
5. What is your budget for security testing?

Justification:

1. Objective: Understanding the objective of the information system will help determine the appropriate testing method. For example, if the system is critical to the business operations and handles sensitive data, penetration testing may be more appropriate.
2. Security requirements: The level of security required will help determine the scope and depth of the testing. If the system requires a high level of security, a more comprehensive testing method like penetration testing may be necessary.
3. Business criticality: The criticality of the information system to the business operations will help determine the level of testing required. If the system is critical to the business, a more thorough testing method like penetration testing may be necessary.
4. Expertise and resources: Understanding the client's current level of security expertise and resources will help determine the appropriate testing method. If the client has limited security expertise and resources, a less complex testing method like fuzz testing may be more appropriate.
5. Budget: Budget is an important consideration when selecting a testing method. Penetration testing is typically more expensive than fuzz testing, so understanding the client's budget will help determine the appropriate testing method.

By asking these questions, the security professional can gain a better understanding of the client's needs and requirements, and make a recommendation on the appropriate testing method (penetration testing or fuzz testing) based on the client's unique situation.

SAGE

Questionnaire for Choosing Between Penetration and Fuzz Testing:

1. What is the intended purpose of the information system?
2. What are the potential security risks or vulnerabilities that you are most concerned about?
3. What is the level of expertise of your development team in terms of security testing?
4. What is the timeline for completing security testing for the information system?
5. What is your budget for security testing?

Justification:

These questions are designed to gather key information that will help me make a recommendation between penetration and fuzz testing. The first question helps to identify the intended purpose of the information system and the potential risks that need to be tested. The second question helps to prioritize the risks and vulnerabilities that require the most attention. The third question helps to determine the level of expertise of the development team in terms of security testing, which will help determine which testing method is more appropriate. The fourth question helps to identify the timeline for completing security testing and whether a more comprehensive or quicker testing method is necessary. The final question helps to

determine the budget for security testing, which will impact the type and scope of testing that can be performed.

****