CISSP Domain 8 – Software Development Security Practice Test (Sample)

Study Guide

Everything you need from our exam experts!

Questions

1. What methodology focuses on the authorized movement and execution of data within a system?

A. Data-centric Threat Modeling

B. DevOps

C. Dynamic Application Security Testing (DAST)

D. Encapsulation

2. Which stage in the traditional software development lifecycle comes after requirements definition?

A. Sprint Planning

B. Testing

C. Systems Design

D. Deployment

3. What term is used to describe security designs utilizing object-oriented programming characteristics?

A. Object Security

B. System Security

C. Object-oriented Security

D. Functional Security

4. Continuous Integration and Continuous Delivery (CI/CD) aims to improve what aspect of software development?

A. System Performance

B. Deployment Speed

C. Data Accuracy

D. Data Recovery

5. What form of programming focuses on the sequence of operations rather than data structures?

A. Object-Oriented Programming

B. Functional Programming

C. Procedural Programming

D. Declarative Programming

6. What is one goal of data classification in software security?

- **A. To develop new data structures**
- **B. To apply appropriate security measures based on data sensitivity**
- **C. To reduce data storage costs**
- **D. To improve data retrieval times**

7. What is the primary purpose of Configuration Management (CM)?

- **A. To reduce development costs**
- **B. To enhance data analysis**
- **C. To maintain system integrity**
- **D. To improve user experience**

8. What is the definition of Advanced Persistent Threats (APTs)?

- **A. Short-term attacks with limited goals**
- **B. Highly sophisticated attacks over extended periods**
- **C. Automated attacks using bots**
- **D. Common hacking attempts by individuals**

9. What does the 'allowed list' refer to in security terms?

- **A. A list of prohibited software**
- **B. A catalog of verified applications and users**
- **C. A database of malware signatures**
- **D. A guide for software development standards**

10. What type of testing seeks to find vulnerabilities by sending incorrect input to a system?

- **A. Static analysis**
- **B. Dynamic analysis**
- **C. Protocol fuzzing**
- **D. Unit testing**

Answers

1. A
2. C
3. C
4. B
5. C
6. B
7. C
8. B
9. B
10. C

Explanations

1. What methodology focuses on the authorized movement and execution of data within a system?

A. Data-centric Threat Modeling

B. DevOps

C. Dynamic Application Security Testing (DAST)

D. Encapsulation

The methodology that focuses on the authorized movement and execution of data within a system is data-centric threat modeling. This approach emphasizes understanding how data flows through a system, identifying potential threats at each stage, and ensuring that appropriate controls are in place to protect data. By modeling data-centric threats, organizations can proactively identify vulnerabilities that may arise from unauthorized access or data manipulation, making it essential for securing sensitive information. Data-centric threat modeling differentiates itself by focusing specifically on the data rather than just the applications or infrastructure where the data resides. This perspective allows for a more granular analysis of how data is used, processed, and stored, and helps in aligning security measures with business objectives and compliance requirements. In contrast, other methodologies mentioned do not focus directly on the management of data movement and execution. For instance, DevOps aims to integrate development and operations to streamline software delivery and does not specifically address data management. Dynamic Application Security Testing (DAST) is a testing approach that evaluates the security of an application during runtime but isn't primarily centered on data flow. Encapsulation refers to an object-oriented programming principle that hides the internal state and behavior of an object but does not address data movement within a system context. Thus, the focus on authorized data handling distinctly aligns data-centric

2. Which stage in the traditional software development lifecycle comes after requirements definition?

A. Sprint Planning

B. Testing

C. Systems Design

D. Deployment

The stage that follows requirements definition in the traditional software development lifecycle is the systems design phase. During this phase, the project team translates the documented requirements into a detailed system architecture and design. This includes defining how the system will operate, specifying the hardware and software requirements, and outlining the overall system structure. The systems design phase is crucial as it sets the foundation for the next stages of development, helping ensure that all functional and non-functional requirements identified earlier are addressed adequately. In contrast, sprint planning refers to agile methodologies where teams plan tasks for the next iteration, not typically a phase following requirements gathering in the traditional model. Testing occurs much later in the process, focusing on validating and verifying the software against requirements. Deployment is the final stage where the completed system is delivered to users, which also comes well after the design phase. Thus, the systems design phase is a key step immediately succeeding requirements definition in the software development lifecycle.

3. What term is used to describe security designs utilizing object-oriented programming characteristics?

A. Object Security

B. System Security

C. Object-oriented Security

D. Functional Security

The term used to describe security designs that utilize object-oriented programming characteristics is known as Object-oriented Security. This approach leverages the principles of object-oriented design, such as encapsulation, inheritance, and polymorphism, to enhance security measures within software development. Object-oriented Security means securing applications by taking advantage of these programming characteristics. For example, encapsulation allows for better data protection, as it restricts direct access to an object's data, allowing only specified methods to interact with that data, which can reduce the likelihood of unauthorized access or manipulation. Additionally, inheritance can enable security features to be reused across different classes, promoting a more consistent security model throughout the program. Polymorphism can provide flexibility in security enforcement by allowing security methods to be overridden in subclasses. This specialized focus on security within the context of object-oriented programming highlights the importance of integrating security into the design and structure of software from the outset, rather than treating it as an afterthought. Such integration leads to a more robust and maintainable security posture in the software development lifecycle. In contrast, terms like Object Security, System Security, and Functional Security are broader concepts that do not specifically address the nuances of object-oriented programming characteristics in the security design context.

4. Continuous Integration and Continuous Delivery (CI/CD) aims to improve what aspect of software development?

A. System Performance

B. Deployment Speed

C. Data Accuracy

D. Data Recovery

Continuous Integration and Continuous Delivery (CI/CD) primarily focus on improving the deployment speed of software applications. By automating the integration of code changes from multiple contributors and streamlining the deployment process, CI/CD reduces the time taken to deliver software updates, fixes, or new features to the end users. This allows developers to release updates more frequently and reliably, thereby enabling faster feedback cycles and the ability to adapt to changes in requirements more quickly. The CI/CD pipeline incorporates automated testing, which ensures that any code integrated into the system maintains the desired quality standards and functionality. As a result, the overall development process becomes more efficient, facilitating a faster time to market without sacrificing quality. This agile approach not only enhances deployment speed but also encourages collaboration among teams, as they can integrate and deploy their changes more seamlessly. While aspects like system performance, data accuracy, and data recovery are important in the broader context of software development and security, they are not the primary focus of CI/CD practices. Instead, CI/CD is specifically designed to optimize how quickly and effectively software can be developed and delivered.

5. What form of programming focuses on the sequence of operations rather than data structures?

A. Object-Oriented Programming

B. Functional Programming

C. Procedural Programming

D. Declarative Programming

Procedural programming is a programming paradigm that emphasizes the procedure or routine to complete tasks and focuses on the sequence of operations executed in order. This style of programming is centered around the concept of procedure calls, where code is organized into reusable blocks or procedures. In procedural programming, the program is divided into discrete functions or procedures that manage the flow of control through the software. The primary focus is not on the data structures themselves but rather on the step-by-step process required to manipulate that data. This allows developers to think in terms of sequences of commands or operations and to structure their code accordingly. By contrast, object-oriented programming revolves around the concept of objects and classes, functional programming emphasizes the application of functions and minimizes side effects, while declarative programming focuses on describing what the program should accomplish rather than how to achieve it through a specific sequence of steps. Each of these paradigms serves a different purpose and approaches problem-solving from unique angles, but procedural programming distinctly prioritizes the flow of operations, making it the correct answer to the question posed.

6. What is one goal of data classification in software security?

A. To develop new data structures

B. To apply appropriate security measures based on data sensitivity

C. To reduce data storage costs

D. To improve data retrieval times

The goal of data classification in software security primarily revolves around applying appropriate security measures based on the sensitivity of the data. By categorizing data into different classes, such as public, confidential, or classified, organizations can determine the level of access control, encryption, and other protective measures that need to be implemented to safeguard the data. This process is critical for ensuring that sensitive information is adequately protected against unauthorized access and potential breaches, while also ensuring that less sensitive data is not overprotected, which could lead to unnecessary expenses and inefficiencies. Effective data classification enables organizations to prioritize security resources and comply with regulatory requirements, ultimately fostering a more secure software environment. The other choices, while they may relate to data management in some capacity, do not directly address the security aspect that data classification aims to achieve. Developing new data structures focuses on the organization or storage of data rather than its security considerations. Reducing storage costs and improving data retrieval times are both operational goals that can be influenced by data management strategies but are not specific to the principles of data classification within a security context.

7. What is the primary purpose of Configuration Management (CM)?

A. To reduce development costs

B. To enhance data analysis

C. To maintain system integrity

D. To improve user experience

The primary purpose of Configuration Management (CM) is to maintain system integrity throughout the software development lifecycle. CM involves systematically managing changes to the system's design, functionality, and configuration in a consistent manner. This ensures that all components are accurately documented and can be tracked, making it easier to manage systems as they evolve over time. Maintaining system integrity is critical, as it helps prevent unauthorized changes that could compromise the security, performance, and reliability of software applications. By establishing processes and tools to automatically track changes and maintain baseline configurations, organizations can ensure continuity, minimize risks, and assist in troubleshooting and recovery operations when issues arise. While reducing development costs, enhancing data analysis, or improving user experience may be associated with effective software development practices, they do not encapsulate the core objective of Configuration Management. Instead, these aspects are indirect benefits that may arise from a well-implemented CM process, which fundamentally focuses on preserving the integrity of the system configurations and components.

8. What is the definition of Advanced Persistent Threats (APTs)?

A. Short-term attacks with limited goals

B. Highly sophisticated attacks over extended periods

C. Automated attacks using bots

D. Common hacking attempts by individuals

Advanced Persistent Threats (APTs) are characterized by their highly sophisticated nature and the prolonged duration of the attacks. These threats typically involve targeted and continuous efforts by attackers to compromise a network or system, aiming to gather sensitive information or maintain access to the environment over an extended timeframe. This distinguishes APTs from other types of attacks, which might be more opportunistic or short-lived. APTs usually involve a deep understanding of the targeted organization and employ multiple tactics to infiltrate, maintain persistence, and eventually extract valuable data. The long-term aspect allows attackers to adapt their strategies in response to the organization’s defenses, making them particularly dangerous. In contrast, the other options describe different attack methodologies or characteristics that do not align with the structured and prolonged nature of APTs. Short-term attacks may involve rapid strikes but lack the persistent aspect. Automated attacks could be extensive but are often less personalized and do not represent the advanced, tailored approach of APTs. Common hacking attempts might consist of opportunistic exploitation rather than the sustained and sophisticated tactics associated with APTs.

9. What does the 'allowed list' refer to in security terms?

A. A list of prohibited software

B. A catalog of verified applications and users

C. A database of malware signatures

D. A guide for software development standards

The 'allowed list' in security terms refers to a catalog of verified applications and users that are permitted to execute or access resources within a system. This approach is pivotal in access control strategies, particularly in managing applications and ensuring that only software that has been vetted for security risks is allowed to run in an environment. By maintaining an allowed list, organizations can effectively mitigate risks associated with unknown or untrusted applications, thus enhancing overall security posture. This concept serves as a proactive measure to control the software environment, generally implemented in conjunction with other security mechanisms, such as intrusion detection systems or anti-malware solutions. It contrasts with a blacklist approach, where the emphasis is on preventing known malicious items while potentially allowing unknown or unverified items, which could also pose risks. The allowed list is thus crucial for minimizing the attack surface and ensuring compliance with security policies.

10. What type of testing seeks to find vulnerabilities by sending incorrect input to a system?

A. Static analysis

B. Dynamic analysis

C. Protocol fuzzing

D. Unit testing

The testing method designed to identify vulnerabilities by sending incorrect or unexpected input to a system is known as protocol fuzzing. This technique involves generating random, unexpected, or malformed data to assess how the system responds to atypical input scenarios. The primary goal of protocol fuzzing is to discover security weaknesses, crashes, memory leaks, or erroneous behavior that may not be apparent through standard testing procedures. By using fuzz testing, security professionals can ensure that systems are robust and resilient against a variety of inputs, which attackers might exploit. This type of testing is particularly important in identifying issues that could lead to security breaches or denial of service attacks. Other testing methodologies, such as static analysis or dynamic analysis, serve different purposes. Static analysis focuses on code examination without executing the program, which helps locate potential vulnerabilities in the source code itself, while dynamic analysis evaluates the program behavior during execution but does not specifically target incorrect input handling as fuzzing does. Unit testing is aimed at verifying the functionality of specific components of the code and is not designed to intentionally produce erroneous input to test system resilience. Therefore, protocol fuzzing stands out as the most aligned with the goal of uncovering vulnerabilities through inappropriate input.