**Security Skills Evaluation Quiz**

|  |  |  |  |
| --- | --- | --- | --- |
| # | Question | Answer | Domains |
| 1 | You are writing a software function that processes external user input. You should:   1. Trust the input as you should always do what the user says even if it is not right. 2. Trust the input as the user always knows what they are doing. 3. Treat the input with suspicion and validate it. 4. Immediately pass it to another function so you don't have to process it. |  | Secure Design Principles  Secure Coding |
| 2 | You are writing software that needs access to multiple platform resources. To make development easier you designed the software to run with root (super user) privileges. Is this the right thing to do?   1. This is OK because you know your software isn't dangerous. 2. This is OK because your software is a special. 3. This is probably a violation of 'defense in depth' principle and you should add more software. 4. This is probably a violation of 'least privilege' principle and you should look at reducing a privilege with which your software runs. |  | Secure Design Principles  Secure Coding  Threat Modeling |
| 3 | You are going to deploy your software into production. You decide that just to be on the safe side you will keep the debug interfaces on even though they are not part of the product requirements. Is this the right thing to do?   1. Yes. You should be rewarded for being proactive. 2. It is all right because those interfaces required authentication. 3. It isn't a problem that the debug interfaces were never validated. 4. Deploying functionality that isn't part of the product requirements potentially increases the attackable surface area. |  | Secure Design Principles  Secure Coding  Threat Modeling  Security Validation |
| 4 | Which one of the following is the most common purpose of threat modeling?   1. Testing an application under realistic scenarios. 2. Identifying potential security threats and their counter measures. 3. Understanding an application's internal data and logic. 4. Performing function testing by using the expected inputs derived from use cases. |  | Threat Modeling |
| 5 | Following code snippet is an example of? (Select all that apply)  char buffer[4];  strcpy(buffer, “SAFECOPY”);   1. Use of an unsafe function 2. Buffer overflow 3. Buffer underflow 4. Input validation |  | Secure Coding  Security Validation |
| 6 | Which of the following statements about buffer overruns are true? (Select all that apply)   1. Buffer overruns occur when data exceeds the expected size and overwrites other values 2. Exists primarily in unmanaged C/C++ code 3. Can be exploited by malware (e.g., rootkits, viruses) 4. Buffer overruns are sometimes called buffer overflows |  | Secure Coding  Security Validation |
| 7 | Which of the following are security properties of cryptographic hash functions? (Select all that apply)   1. Fixed output length 2. Collision resistance 3. One way-ness or preimage resistance 4. Relatively easy to compute |  | Cryptography |
| 8 | Which of the following cryptographic algorithms are symmetric key algorithms?   1. ECDSA 2. RSA 3. AES 4. DES 5. RC4 |  | Cryptography |
| 9 | An organization has developed an application that manages highly sensitive information about its users, such as their credit card and banking information. The organization has developed its own private cryptographic library to protect the data and has asked you with reviewing its implementation. How would you advise this organization?   1. This is not recommended. You should avoid developing your own cryptographic library. 2. This is recommended. The more tailored the crypto implementation is to the individual application the better protected your data will be. 3. This is recommended. Since the crypto implementation is kept secret, attackers will be unable to decrypt the protected data. |  | Security Validation  Cryptography |
| 10 | Is it important to consider SDL in the initial project planning phase?   1. The SDL can start later in the process if the Product Program Manager determines, it is better for the product development process. 2. If SDL tasks are not done at the appropriate development time, security flaws might not be discovered until late in the development when they can be more difficult and costly to fix. 3. SDL can start after Alpha when product functionality and risks are more understood. |  | SDL |
| 11 | An electronic water supply valve fails in an open state with water running. Which basic principle of security is violated here:   1. Defense in depth 2. Principle of least privilege 3. Fail safely and securely |  | Secure Design Principles |
| 12 | As pre-SDL requirement, every developer needs core security training in which of the following areas? (Select all that apply)   1. Secure design 2. Threat modeling 3. Secure coding 4. Security testing 5. Privacy |  | SDL |
| 13 | When is the most appropriate time to start the SDL process?   1. After the product is shipped 2. When the first version of software is available 3. After product requirements have been finalized 4. As early as possible (at project planning phase) |  | SDL |
| 14 | When an SDL gap is discovered in a component, does it mean the component can't be used in your product?   1. No, unless the component is from an outside vendor. 2. Yes, any discovered SDL gap which can't be made compliant by the required milestone may not be used in the product. 3. Yes, when the product is used by the financial, medical or government industries as they have their own standards. 4. Not necessarily, not all SDL gaps are showstoppers, some can be low risk and can be quickly addressed. |  | SDL |
| 15 | To avoid security vulnerabilities, which of the following practices you should avoid?   1. Implement your own cryptographic algorithms 2. Store secrets unencrypted on disk 3. Hiding keys or passwords in the code 4. Use Key Derivation Function (KDF) to derive several keys from a secret |  | Secure Coding  Security Validation  Cryptography |
| 16 | Your software is complete and ready to ship you have no known security vulnerabilities. Although you have no data to support it, you are worried that the security protections offered by your compiler might be slowing your software. You decide to re-compile your software without compiler security protections.   1. Removing the compiler security protection is a violation of “least privilege” security principle. 2. Removing the compiler security protections is a violation of “defense in depth” security principle. 3. Removing the compiler security protections is a violation of “fail intelligently” security principle. 4. Removing the compiler security protections is a violation of “secure by default” security principle. |  | Secure Design Principles  Secure Coding  Security Validation |
| 17 | In the following code example, the user is free to specify the quantity of items to be purchased and a total is calculated. The user has no control over the price variable, however the code does not prevent a negative value from being specified for quantity. A negative value for the quantity will result in the user's account to be credited. Which secure coding policy is violated here?  public static final double price = 20.00;  int quantity = currentUser.getAttribute("quantity");  double total = price \* quantity;  chargeUser(total);   1. Validate all input 2. Prevent buffer overflows 3. Prevent race conditions 4. Watch the cryptgraphy code |  | Secure Coding  Security Validation |
| 18 | The following example takes a user-supplied value to allocate an array of objects and then operates on the array. This code attempts to build a list from a user-specified value, and even checks to ensure a non-negative value is supplied. If, however, a 0 value is provided, the code will build an array of size 0 and then try to store a new Widget in the first location, causing an exception to be thrown. Which secure coding policy is violated here?  private void buildList ( int untrustedListSize ){  if ( 0 > untrustedListSize ){  die("Negative value supplied for list size, die evil hacker!");  }  Widget[] list = new Widget [ untrustedListSize ];  list[0] = new Widget();  }   1. Validate all input 2. Prevent buffer overflows 3. Prevent race conditions 4. Principle of least privilege |  | Secure Design Principles  Secure Coding  Security Validation |
| 19 | What does the security acronym CIA stands for:   1. Clarity, Impact, Assessment 2. Confidentiality, Insight, Availability 3. Confidentiality, Integrity, Availability 4. Clarity, Integrity, Application |  | Threat Modeling |
| 20 | Which of the following are security properties of assets? (Select all that apply)   1. Confidentiality 2. Integrity 3. Clarity 4. Availability 5. Impact 6. Access violation 7. Performance 8. Policy violation |  | Threat Modeling |
| 21 | Examples of assets are:   1. A cryptographic key 2. DLL’s of an application 3. Up time of a server 4. A hardware platform's model number |  | Threat Modeling |
| 22 | Match the threats below to their respective mitigation techniques:   |  |  | | --- | --- | | Threat | Mitigation technique | | 1. Spoofing |  | | 1. Tampering |  | | 1. Repudiation |  | | 1. Information Disclosure |  | | 1. Denial of service |  | | 1. Elevation of Privilege |  |   Mitigation techniques:   |  | | --- | | 1. Availability mechanisms | | 1. Authentication | | 1. Authorization controls | | 1. Digital signing, logging | | 1. Applying confidentiality controls | | 1. Integrity verification | |  | Threat Modeling |
| 23 | You developed some firmware for an IOT device. An attacker was able to execute modified firmware. Which security property of the asset is violated here:   1. Confidentiality 2. Integrity 3. Availability 4. Access violation 5. Policy violation |  | Threat Modeling |
| 24 | What is the minimal set of the following elements are required to have a potential threat?   1. Asset 2. Attacker with some capabilities 3. Entry point 4. Working exploit 5. Vulnerability in your code |  | Threat Modeling |
| 25 | Which of the following is an attack point?   1. A trusted element that can be compromised 2. Cryptographic key 3. The entry point into th system for an adversary to begin an attack |  | Threat Modeling |
| 26 | For what reasons would an identified threat not be mitigated? (Choose all that apply)   1. Low impact 2. Low probability 3. Recommendation by program manager to ship the product as is |  | Threat Modeling |
| 27 | The Principle of Least Privilege says that an object: (choose all that apply)   1. Should be given no more privilege than is required to perform its specific function. 2. Can access only information and resources that are necessary for their proper operation. 3. Has permissions only for as long a necessary to get its job done. 4. Requires super-user privilege to run, to make access control logic simpler, thus reducing the attack surface. |  | Secure Design Principles |
| 28 | What are the objectives in the application of the Principle of Least Privilege? (choose all that apply)   1. Decreasing the risk of compromising a highly privileged entity giving access to critical assets. 2. Decreasing the attack surface for malicious software. 3. To provide documentation that you are following security principles in your development. 4. Providing modules with administrative permissions for ease of development. |  | Secure Design Principles |
| 29 | What mechanism can you employ if your component needs access to protected resources and can't or shouldn't elevate?   1. Man-in-the middle 2. Broker 3. Interceptor 4. Secret interface |  | Secure Design Principles |
| 30 | At what point in the product lifecycle should the Principle of Least Privilege be applied? (choose all that apply)   1. In planning, refresh and development 2. During design, consider where applications, services, and libraries will run 3. In validation, ensure we have not over privileged components or opened attack surfaces. 4. Only when product is ready to be shipped. |  | Secure Design Principles  Security Validation |
| 31 | Which of the following types of bugs are detected by fuzzing? (Select that all apply)   1. Failures to detect &/or handle errors and exceptions 2. Memory corruption 3. Arithmetic errors in pointer addresses and array indexes leading to crashes 4. Memory leaks (in combination with other tools) 5. Race conditions (in combination with other tools) |  | Security Validation  Security assurance methods and tools |
| 32 | Which of the following statements about smart fuzzing are true? (Select that all apply)   1. Structure-aware (Fuzzing data is modeled on the expected input structure) 2. Takes much more time and effort to implement compared to dumb fuzzing 3. Smart fuzzing scratches the surface and exposes “shallow” bugs |  | Security Validation  Security assurance methods and tools |
| 33 | Which of the following statements about dumb fuzzing are true? (Select that all apply)   1. Dumb fuzzing is structure aware. 2. Relatively quick and easy to implement. 3. Limited code/interface coverage (Finds the “low-hanging” fruit.) |  | Security Validation  Security assurance methods and tools |
| 34 | Which interfaces are likely candidates for fuzzing? (Select that all apply)   1. Inputs to APIs 2. Interrupts/events 3. File parser inputs 4. GUIs/Command line options 5. If it is receiving data from somewhere, it can be fuzzed. |  | Security Validation  Security assurance methods and tools |
| 35 | True or false?  Mutation based fuzzing Starts with a sample of valid input |  | Security Validation |
| 36 | The OpenSSL Heartbleed vulnerability potentially allows an attacker to read a large portionof server memory. This vulnerability represents a breakdown in:   1. Least privilege 2. Availability 3. Confidentiality 4. Integrity |  | Threat Modeling  Security Validation |
| 37 | An easy to find vulnerability in an application developed by your organization was just publicly published on the Internet along with an exploit tool. Your development team is addressing that particular vulnerability right now, but as the stakeholder you need to quickly identify other vulnerabilities to reduce the risk to customers. You don’t have in-house security expertise and the code base is rather large so what can you do?   1. Use a static code analysis tool 2. Use input validation 3. Use manual code reviews |  | Secure Coding  Security Validation  Security assurance methods and tools |
| 38 | Which of the following are secure coding practices? (Select that all apply)   1. Validate all input 2. Avoid arithmentic errors 3. Optimize for memory space 4. Prevent buffer overflows 5. Optimize for speed |  | Secure Coding  Security Validation |
| 39 | Which one of the following best describes the black-box fuzzing process?   1. Named resources and objects that might be shared by the application are identified. 2. Malformed data is sent to the application after ensuring that all target code that will be tested. 3. Extraordinary conditions that the developers and designers might not have considered in an application are simulated. 4. Malformed data is sent to the application without verifying which code paths will be tested. |  | Security Validation  Security assurance methods and tools |
| 40 | Attackers commonly exploit buffer overflows to attack software. A buffer overflow allows an attacker to write into a memory location unexpected by the developer. This vulnerability represents a breakdown in:   1. Hashing 2. Integrity 3. Attackable surface area 4. Availability |  | Secure Coding  Threat Modeling  Security Validation |
| 41 | Which of the following can help mitigate buffer overflows? (Select that all apply)   1. Off-by-one 2. Bound checking functions 3. Regular expressions 4. Raw pointers |  | Secure Coding  Security Validation |
| 42 | Which one of the following is the most common objective of memory corruption vulnerability exploit? (Select that all apply)   1. Create logic errors 2. Create arithmetic errors 3. Arbitrary code execution |  | Secure Coding  Threat Modeling  Security Validation |
| 43 | Which of the programming language features will you use to help mitigate issues with manual memory management? (Select that all apply)   1. Regular expressions 2. Raw pointers 3. Alternative to dangerous functions |  | Secure Coding |
| 44 | Memory leaks can cause the following? (Select that all apply)   1. Denial of service when leaks occur repeatedly 2. Off-by-one indexing errors 3. Functional errors |  | Secure Coding  Threat Modeling  Security Validation |
| 45 | What is the vulnerability in the code below?  bool is\_file\_accessible(const char \*fname) {  return (strncmp(“/tmp/”, fname, strlen(“/tmp/”)) == 0);  }   1. Buffer overflow 2. Out-of-bounds read 3. Path traversal 4. Integer overflow |  | Secure Coding  Security Validation |
| 46 | What is the vulnerability in the code below?  def print\_message(msg):  os.system(‘echo {}’.format(msg))   1. Command injection 2. Heap corruption 3. Indexing error |  | Secure Coding  Security Validation |
| 47 | Which method of handling input from files is considered secure? (Select that all apply)   1. Read the file into memory and validate it. When data is needed, read it again 2. Read the file into memory and validate it. When data is needed, use the validated copy 3. Lock the file for writing and validate its contents. When data is needed, read it from the locked file |  | Secure Coding  Security Validation |
| 48 | What tool can help you identify missing or incorrectly configured compiler/linker security flags? (Select that all apply)   1. Klocwork 2. Checksec 3. Black Duck 4. Fuzzer |  | Security Validation  Security assurance methods and tools |