PassQueen
a.k.a dumpsqueen.com

BTA Certified Blockchain Security Professional

Blockchain CBSP

Version Demo

Total Demo Questions: 11

Total Premium Questions: 116

Buy Premium PDF

https://passqueen.com

support@passqueen.com

QUESTION NO: 1

Which of the following controls help reduce the risk of private-key loss for an organizational cryptocurrency treasury? Select all that apply.

A. Threshold authorization

B. Offline backup procedures

C. Documented key-recovery processes

D. Unprotected shared cloud storage

ANSWER: A B C

Explanation:

Threshold authorization, **offline backup procedures**, and **documented key-recovery processes** are appropriate controls. A threshold authorization arrangement, such as an m-of-n multisignature wallet, prevents one individual or one compromised private key from independently transferring treasury assets. It also improves resilience when an authorized signer is unavailable.

Offline backups of recovery material, protected against unauthorized access and environmental loss, help an organization recover access after device failure or loss. A documented recovery process defines the authorized personnel, approval requirements, storage locations, and verification steps needed to restore access without improvisation during an incident. Storing a recovery phrase in an unprotected shared cloud folder creates a high risk of unauthorized copying and theft rather than reducing key-loss risk. See the Bitcoin Developer Guide: Wallets and the NIST SP 800-57 Part 1 for key-management guidance.

QUESTION NO: 2

Which types of network attacks focus on partitioning the blockchain network? Select all that apply

A. Eclipse

B. Sybil

C. Denial of Service

D. Routing

ANSWER: A D

Explanation:

Eclipse and **Routing** attacks directly focus on partitioning or isolating parts of a blockchain peer-to-peer network. An eclipse attack surrounds a target node with attacker-controlled peers, controlling the node's view of transaction and block propagation. This effectively partitions that node from honest peers, allowing the attacker to delay, filter, or selectively provide blockchain information. The Bitcoin peer-to-peer model depends on nodes maintaining diverse connections and independently relaying inventory and data, so monopolizing a node's connections defeats that diversity. See the Bitcoin Developer Guide's peer-to-peer network overview and the foundational Eclipse Attacks on Bitcoin's Peer-to-Peer Network paper.

Routing attacks target the Internet-layer paths that blockchain nodes use to communicate. For example, manipulation or hijacking of routing announcements can separate groups of nodes, prevent block and transaction propagation between them, or isolate a region or autonomous system. The intended security impact is a network partition: participants on each side have incomplete and potentially inconsistent views of the current chain until connectivity is restored. Both attack types therefore directly address network connectivity and information flow, which are the core mechanisms involved in partitioning a blockchain network.

QUESTION NO: 3

Which of the following attacks only requires a single account and performs only normal blockchain operations?

A. Replay Attack

B. Routing Attack

C. 51% Attack

D. Eclipse Attack

ANSWER: A

Explanation:

Replay Attack is correct because it can be performed by taking a transaction that was validly signed and broadcast previously and submitting the same signed transaction again in another context where it remains valid. The attacker does not need to control network routing, operate a majority of consensus resources, or compromise other participants. Instead, the attack relies on ordinary protocol behavior: nodes receive, validate, and relay transactions that have valid signatures and satisfy the target chain's transaction rules.

This risk is especially important when two compatible chains share transaction-signing formats or when an application fails to bind an authorization to a unique context. Replay protection addresses the issue by incorporating context-specific data into the signed transaction, such as a chain identifier, and by using transaction nonces to ensure that each transaction from an account is processed only once on a particular chain. Ethereum's EIP-155 specifies chain-ID-based replay protection for signed transactions, while Ethereum's transaction documentation explains the role of nonces in ordering and uniqueness. See EIP-155: Simple replay attack protection and Ethereum transaction documentation.

QUESTION NO: 4

Multisignatures are designed to allow a set of users to make a valid transaction only if a set minimum number of them consent

A. True

B. False

ANSWER: A

Explanation:

True is correct. A multisignature arrangement, commonly expressed as *m-of-n*, requires signatures from at least a specified minimum number (*m*) of the total authorized signers (*n*) before funds can be spent or a transaction can be authorized. For example, a 2-of-3 multisignature wallet has three authorized key holders, but any two of them must provide valid signatures to create a spend that satisfies the locking conditions. This design distributes authorization authority rather than leaving control with one private key holder. It is useful for organizational treasuries, shared custody, escrow-like processes, and resilience planning, because transaction authorization can continue when one signer is unavailable while still requiring collective approval. In Bitcoin, Pay-to-Script-Hash and Segregated Witness script forms can encode multisignature spending conditions, and modern descriptor syntax can represent them with functions such as `multi()` or `sortedmulti()`. The essential characteristic is therefore the threshold: a transaction becomes valid only when the required minimum number of authorized participants have consented by supplying valid signatures. See the Bitcoin Developer Guide: Transactions and Bitcoin Core output script descriptors documentation.

QUESTION NO: 5

Which of the following attacks takes advantage of the fact that transaction information is posted on the blockchain to infer sensitive information?

A. Smart Contract Exploitation

B. Data Mining

C. Brute Force Key Guessing

D. Cryptanalysis

ANSWER: B

Explanation:

Data Mining is correct because public blockchains make transaction records broadly available for inspection, creating a rich dataset that can be analyzed to identify patterns and infer information that is not explicitly written in a transaction. An attacker or analyst can correlate transaction amounts, timing, address reuse, transaction-graph relationships, and interactions with known services to cluster addresses that may belong to the same entity. Those clusters can then be associated with individuals or organizations when combined with external information, such as exchange records, public donation addresses, merchant data, or information disclosed elsewhere.

This is a privacy risk because pseudonymous addresses do not necessarily provide anonymity. Although a blockchain transaction may not directly contain a person's name, systematic analysis of public ledger data can reveal behavioral, financial, and relationship information. The Bitcoin developer documentation specifically notes that the public transaction history can expose privacy-sensitive information and recommends avoiding address reuse. The National Institute of Standards and Technology likewise describes how blockchain data can be analyzed and linked with off-chain information. These characteristics make data mining and transaction-graph analysis a practical method for inferring sensitive details from publicly posted blockchain transactions. See the Bitcoin Developer Guide on privacy and NISTIR 8202, Blockchain Technology Overview.

QUESTION NO: 6

All smart contracts are audited for correctness and checked for malicious code before being uploaded to the blockchain.

A. True

B. False

ANSWER: B

Explanation:

False is correct because public blockchain networks do not impose a universal, automatic review process that proves every smart contract is correct or free of malicious behavior before deployment. A contract creator can generally submit deployment transactions directly to the network, provided the transaction meets the protocol's validity requirements, such as sufficient fees and valid authorization. Network validation establishes that a transaction follows consensus rules; it does not establish that the deployed program's business logic is safe, intended, or non-malicious.

Audits, formal verification, testing, and code review are important security practices, but they are normally arranged by the contract's developers, project owners, users, or independent security firms. Their scope and quality vary, and even a completed audit is not an absolute guarantee that a contract has no vulnerabilities. This distinction is fundamental to blockchain security: once deployed, contract code may execute exactly as written, including unintended logic or deliberately harmful functionality. Ethereum's security guidance emphasizes that developers must design and test contracts carefully, while its documentation describes deployment as a transaction submitted to the network rather than a mandatory audit workflow. See Ethereum smart-contract security guidance and Ethereum deployment documentation.

QUESTION NO: 7

Which of the following blockchains is designed to allow multiple blockchains to run on the same network?

A. Hyperledger

B. Ethereum

C. Corda

ANSWER: A

Explanation:

Hyperledger is correct in this context, specifically through Hyperledger Fabric's channel architecture. A Fabric network can be partitioned into channels, where each channel provides a separate, private communication path and maintains its own ledger. This means organizations can use the same underlying peer network and ordering infrastructure while operating multiple logically distinct blockchain ledgers. Only the organizations authorized for a particular channel can access that channel's transactions, smart contracts, and ledger data, supporting confidentiality among different business groups operating within a shared consortium environment.

This architecture is useful when several workflows or groups require independent records and privacy boundaries but still benefit from common network administration and shared infrastructure. Each channel has a distinct blockchain ledger, and peers may join one or more channels according to the data and transaction flows in which they participate. Hyperledger Fabric documentation describes channels as private "subnets" of communication between specific network members and confirms that each channel has a separate ledger. See the Hyperledger Fabric Channels documentation and the Hyperledger Fabric Ledger documentation.

QUESTION NO: 8

Malicious smart contracts can potentially infect the nodes running the blockchain software

A. True

B. False

ANSWER: B

Explanation:

False is correct because smart-contract code is designed to execute inside a blockchain virtual machine rather than as unrestricted native code on the host operating system. For example, Ethereum contract bytecode runs in the Ethereum Virtual Machine, a deterministic, sandboxed execution environment replicated by participating nodes. A contract can alter only blockchain state according to protocol rules and the permissions encoded in the contract; it does not normally receive direct access to a node's operating system, filesystem, processes, or network interfaces. Therefore, deploying or calling a malicious smart contract does not itself "infect" blockchain nodes in the sense of installing malware or propagating host-level malicious software.

Node operators must still apply security best practices, including keeping client software patched and isolating node infrastructure. A vulnerability in a particular node client or its surrounding environment could potentially be exploited by specially crafted data, but that is a software implementation vulnerability, not an inherent capability of smart contracts. The expected security boundary is the virtual-machine sandbox. Ethereum describes the EVM as the environment used to execute smart-contract code: Ethereum Virtual Machine documentation. Solidity's security guidance also discusses the execution and security model for contracts: Solidity Security Considerations.

QUESTION NO: 9

Which of the following smart contract vulnerabilities can cause a smart contract's balances ledger to become incorrect? Select all that apply

A. Arithmetic

B. Unchecked Return Values

C. Race Condition

D. Reentrancy

ANSWER: A B C D

Explanation:

Arithmetic, unchecked return values, race conditions, and reentrancy can each corrupt the logical accounting represented by a smart contract's balances ledger. Arithmetic errors—especially overflow or underflow in Solidity versions before built-in checked arithmetic—can cause a credit or debit calculation to wrap to an unintended value. Unchecked return values can create an accounting mismatch when a contract records a transfer, payout, or token movement as completed even though the external call or token operation failed. Race conditions, including transaction-ordering dependencies, can allow multiple pending actions to be evaluated against stale or unexpected state, resulting in balance updates that do not reflect the intended sequence of operations. Reentrancy is particularly dangerous for balance management: an external recipient can call back into a vulnerable contract before its balance is reduced, potentially triggering repeated withdrawals while the ledger still shows funds as available. Secure balance handling therefore requires checked arithmetic, explicit success checks, carefully ordered state transitions, and the checks-effects-interactions pattern or reentrancy protection. See the Solidity security considerations and the Smart Contract Weakness Classification Registry for recognized weakness patterns and mitigations.

QUESTION NO: 10

Which of the following features were exploited as part of the Verge cryptocurrency hack?

Select all that apply.

A. Mining Difficulty Update Function

B. Multiple Hash Functions

C. Timestamp Flexibility

D. Short Addresses

ANSWER: A B C

Explanation:

The Verge attacks exploited weaknesses involving the **Mining Difficulty Update Function**, **Multiple Hash Functions**, and **Timestamp Flexibility**. Verge supported multiple proof-of-work hashing algorithms, with difficulty handling intended to accommodate mining across those algorithms. The attacker manipulated block-time information and exploited the interaction between timestamp validation and difficulty retargeting. By supplying timestamps that made blocks appear to have been produced at abnormal intervals, the attacker induced an incorrectly low mining difficulty for a selected algorithm. This allowed blocks to be generated far more rapidly than the network's intended block schedule and enabled the attacker to receive an outsized share of block rewards. This is a classic example of a time-warp/difficulty-adjustment failure: consensus-critical timestamp rules and retarget calculations must be designed together, bounded carefully, and tested against adversarial inputs. Multi-algorithm proof-of-work networks require particular care because each algorithm's work and difficulty accounting must remain consistent and resistant to cross-algorithm manipulation. Verge's source repository and its historical fixes provide useful primary context for the project's consensus implementation: Verge GitHub repository. Additional technical reporting on the timestamp-based exploitation and rapid block generation is available from CoinDesk's report on the Verge attack.

QUESTION NO: 11

Which of the following are appropriate controls for reducing replay risk after a blockchain hard fork? Select all that apply.

A. Chain-specific transaction signing

B. Replay-protection rules

C. Separating funds before transacting on both chains

D. Reusing identical signed transactions on both chains

ANSWER: A B C

Explanation:

Chain-specific transaction signing, **replay-protection rules**, and **separating funds before transacting on both chains** are correct. A hard fork can produce two networks that recognize the same pre-fork transaction history and keys. Without replay protection, a transaction validly signed for one chain may also be valid on the other chain, allowing an unintended duplicate transfer.

Chain-specific signing binds a signature to a designated network or chain identifier. Protocol-level replay-protection rules can make transactions on one branch invalid on the other. Users may also separate or split funds through a carefully designed transaction process before making independent payments on each chain. Reusing identical signed transactions on both branches increases, rather than reduces, replay exposure. Ethereum's EIP-155 introduced chain identification into transaction signing to provide replay protection across chains. See EIP-155: Simple replay attack protection.