Secureum Bootcamp Ethereum 101 Quiz

October 18, 2021 / patrickd

This is a writeup of the Secureum Bootcamp Ethereum 101 Quiz containing solutions and references to the provided study material.
For fairness it was published after submissions to it were closed.

The quiz consisted of 32 questions with a strict timelimit of 16 minutes. The ordering of the questions was randomized, so the numbering here won't match with the numbering elsewhere.

> "Which of the following EVM components is/are non-volatile across transactions?"
>
> — 1 of 32

- [ ] A. Stack
- [ ] B. Memory
- [x] C. Storage
- [ ] D. Calldata

▼ Solution

Correct is C.

> A stack machine is a computer processor or a virtual machine in which the primary interaction is moving **short-lived temporary values to and from a push down stack**.

from Stack machine - Wikipedia

> The EVM is a simple stack-based architecture consisting of the stack, **volatile memory**, **non-volatile storage** with a word size of 256-bit (chosen to facilitate the Keccak256 hash scheme and elliptic-curve computations) and Calldata.

from point 59 of Ethereum 101 - by Secureum

> Calldata is a read-only byte-addressable space where the **data parameter of a transaction or call** is held.

from point 63 of Ethereum 101 - by Secureum

> "The number of transactions in a Ethereum block depend on"
>
> — 2 of 32

- [ ] A. Nothing. It is a constant
- [x] B. Gas used by transactions
- [x] C. Block gas limit
- [ ] D. Block difficulty

▼ Solution

Correct is B & C.

> Block gas limit is set by miners and refers to the cap on the total amount of gas expended by all transactions in the block, which ensures that blocks can't be arbitrarily large. Blocks therefore are not a fixed size in terms of the number of transactions because different transactions consume different amounts of gas.

from point 48 of Ethereum 101 - by Secureum

> “EVM Stores”

— 3 of 32

- [x] A. Most significant byte in the smallest memory address
- [ ] B. Most significant byte in the largest memory address
- [x] C. In Big-endian order
- [ ] D. In Little-endian order

▼ Solution

Correct is A & C.

> EVM uses big-endian ordering where the most significant byte of a word is stored at the smallest memory address and the least significant byte at the largest

from point 65 of Ethereum 101 - by Secureum

> “Ethereum’s thread model is characterised by”

— 4 of 32

- [ ] A. Trusted miners and users
- [ ] B. Trusted users, untrusted miners
- [ ] C. Trusted miners, untrusted users
- [x] D. Everyone is untrusted

▼ Solution

Correct is D.

> Given the aspirational absence of trusted intermediaries, everyone and everything is meant to be untrusted by default. Participants in this model include developers, miners/validators, infrastructure providers and users, all of whom could potentially be adversaries.

from point 95 of Ethereum 101 - by Secureum

> “Ethereum smart contracts do not run into halting problems because”
>
> — 5 of 32

- [ ] A. EVM is not Turing Complete
- [ ] B. EVM is Turing Complete
- [x] C. EVM is Turing Complete but is bounded by gas sent in transaction
- [ ] D. EVM is Turing Complete but is bounded by the stack depth

▼ Solution

Correct is C.

> Turing-complete systems face the challenge of the halting problem i.e. given an arbitrary program and its input, it is not solvable to determine whether the program will eventually stop running. So Ethereum cannot predict if a smart contract will terminate, or how long it will run. Therefore, to constrain the resources used by a smart contract, Ethereum introduces a metering mechanism called gas.

from point 10 of Ethereum 101 - by Secureum

> “Which of the following operation(s) touch(es) storage?”
>
> — 6 of 32

- [ ] A. SWAP
- [x] B. SLOAD
- [ ] C. DUP
- [ ] D. PUSH

▼ Solution

Correct is B.

> Most EVM instructions operate with the stack (stack-based architecture) and there are also stack-specific operations e.g. PUSH,

> POP, SWAP, DUP etc.

from point 60 of Ethereum 101 - by Secureum

> Storage is a 256-bit to 256-bit key-value store. [...] This is accessed with SLOAD/SSTORE instructions.

from point 62 of Ethereum 101 - by Secureum

> “The most gas-expensive operation is”

— 7 of 32

- [ ] A. SLOAD
- [ ] B. SSTORE
- [x] C. CREATE
- [ ] D. SELFDESTRUCT

▼ Solution

Correct is C.

> SLOAD is 2100 gas and SSTORE is 20,000 gas to set a storage slot from 0 to non-0 and 5,000 gas otherwise. CREATE is 32000 gas and SELFDESTRUCT is 5000 gas.

from point 78 of Ethereum 101 - by Secureum

> “Transaction T1 attempts to write to storage values S1 and S2 of contract C. Transaction T2 attempts to read the same storage values S1 and S2. However, T1 reverts due to an exception after writing S2. Which or the following is/are TRUE?”

— 8 of 32

- [ ] A. T2 reads the value of S1 updated by T1

- [x] B. T2 reads the value of S1 prior to T1's attempted update
- [ ] C. T2 also reverts because of the dependency on T1
- [ ] D. This scenario is not possible

▼ Solution

Correct is B.

> Transaction properties: Atomic: it is all or nothing i.e. cannot be divided or interrupted by other transactions

from point 32 of Ethereum 101 - by Secureum

> A transaction reverts for different exceptional conditions such as running out of gas, invalid instructions etc. in which case all state changes made so far are discarded and the original state of the account is restored as it was before this transaction executed.

from point 79 of Ethereum 101 - by Secureum

> “The gas tracking website https://etherscan.io/gastracker says that Low gas cost is 40 gwei. This affects”
>
> — 9 of 32

- [x] A. The transaction gasPrice
- [ ] B. The transaction gasLimit
- [ ] C. The transaction value
- [ ] D. Both B & C

▼ Solution

Correct is A.

> Gas price: The price a transaction originator is willing to pay in exchange for gas. The price is measured in wei per gas unit. The higher the gas price, the faster the transaction is likely to be confirmed on the blockchain. The suggested gas price depends on the demand for block space at the time of the transaction.

from point 35 of Ethereum 101 - by Secureum

> gasLimit: The maximum amount of gas the originator is willing to pay for this transaction value: The amount of ether (in wei) to send to the destination

from point 33 of Ethereum 101 - by Secureum

> “Security of Ethereum DApps depend on”

— 10 of 32

- [x] A. Security of their smart contracts
- [x] B. Security of their off-chain components
- [x] C. Security of Ethereum
- [ ] D. None of the above

▼ Solution

Correct is A, B, C.

> On-chain vs Off-chain: Smart contracts are “on-chain” Web3 components and they interact with “off-chain” components that are very similar to Web2 software. So the major differences in security perspectives between Web3 and Web2 mostly narrow down to security considerations of smart contracts vis-a-vis Web2 software.

from point 90 of Ethereum 101 - by Secureum

> “A nonce is present in”

— 11 of 32

- [ ] A. Ethereum transaction
- [ ] B. Ethereum account

- [x] C. Both A & B
- [ ] D. Neither A nor B

▼ Solution

Correct is C.

> Ethereum account contains four fields: The nonce, a counter used to make sure each transaction can only be processed once...

from point 23 of Ethereum 101 - by Secureum

> A transaction is a serialized binary message that contains the following components: nonce: A sequence number, issued by the originating EOA, used to prevent message replay...

from point 33 of Ethereum 101 - by Secureum

> “Miners are responsible for setting”
>
> — 12 of 32

- [ ] A. Transaction gas price
- [x] B. Block gas limit
- [ ] C. Both A & B
- [ ] D. Neither A nor B

▼ Solution

Correct is B.

> Gas price: The price a transaction originator is willing to pay in exchange for gas. The price is measured in wei per gas unit. The higher the gas price, the faster the transaction is likely to be confirmed on the blockchain. The suggested gas price depends on the demand for block space at the time of the transaction.

from point 35 of Ethereum 101 - by Secureum

> Block gas limit is set by miners and refers to the cap on the total amount of gas expended by all transactions in the block, which ensures that blocks can't be arbitrarily large.

from point 48 of Ethereum 101 - by Secureum

> "Which of the following information CANNOT be obtained in the EVM?"
>
> — 13 of 32

- [ ] A. Block difficulty
- [x] B. Transaction logs
- [ ] C. Balance of an account
- [x] D. Block hash of any block

▼ Solution

Correct is B, D.

> 0x31 BALANCE 1 1 Get balance of the given account
> 0x40 BLOCKHASH 1 1 Get the hash of one of the **256 most recent complete blocks**
> 0x44 DIFFICULTY 0 1 Get the block's difficulty

(there's no operation to access transaction logs)

from points 70, 71 of Ethereum 101 - by Secureum

> "Miners are incentivized to validate and create new blocks by"
>
> — 14 of 32

- [x] A. Block rewards
- [ ] B. Altruism
- [x] C. Transaction fees
- [ ] D. Their belief in decentralization

▼ Solution

Correct is A, C.

> Miners are rewarded for blocks accepted into the blockchain with a block reward in ether (currently 2 ETH). A miner also gets fees which is the ether spent on gas by all the transactions included in the block.

from point 47 of Ethereum 101 - by Secureum

> “Smart contracts on Ethereum”
>
> — 15 of 32

- [x] A. May be deployed by anyone
- [ ] B. May be deployed only through the DApp store
- [x] C. May have some form of access control
- [ ] D. Are guaranteed to be secure

▼ Solution

Correct is A, C.

> Web3: is a **permissionless**, trust-minimized and censorship-resistant network for transfer of value and information.

from point 88 of Ethereum 101 - by Secureum

> “Hardfork on Ethereum”
>
> — 16 of 32

- [ ] A. Has never happened
- [ ] B. Happened only once after the DAO attack
- [x] C. Happens with backwards-incompatible protocol changes
- [ ] D. Happens when developers and miners disagree on changes

▼ Solution

Correct is C.

> A hard fork to introduce an exponential difficulty increase, to motivate a transition to PoS when ready....

from "Ethereum's Four Stages of Development" of Mastering Ethereum

> "Which call instruction could be used to allow modifying the caller account's state?"
>
> — 17 of 32

- [ ] A. CALL
- [x] B. CALLCODE
- [x] C. DELEGATECALL
- [ ] D. STATICALL

▼ Solution

Correct is B, C.

> 0xf1 CALL 7 1 Message-call into an account
> 0xf2 CALLCODE 7 1 Message-call into this account with an alternative account's code
> 0xf4 DELEGATECALL 6 1 Message-call into this account with an alternative account's code, but persisting the current values for sender and value
> 0xfa STATICCALL 6 1 Static message-call into an account

from point 77 of Ethereum 101 - by Secureum

> Another variant of call is delegatecall, which replaced the more dangerous callcode. [...] Essentially, delegatecall runs the code of another contract inside the context of the execution of the current contract.

from "Calling Other Contracts (send, call, callcode, delegatecall)" of Mastering Ethereum

> Permits non-state-changing calls to itself or other contracts while disallowing any modifications to state during the call (and its subcalls, if present) to increase smart contract security and assure developers that re-entrancy bugs cannot arise from the call.

from "Appendix A: Ethereum Standards" of Mastering Ethereum

> “The length of addresses on Ethereum is”

— 18 of 32

- [ ] A. 256 bits
- [x] B. 20 bytes
- [ ] C. Depends on the Externally-Owned-Account or Contract address
- [ ] D. Configurable

▼ Solution

Correct is B.

> Ethereum state is made up of objects called "accounts", with **each account having a 20-byte address** and state transitions being direct transfers of value and information between accounts.

from point 22 of Ethereum 101 - by Secureum

> “Which of the following statements is/are TRUE about gas?”

— 19 of 32

- [ ] A. Unused gas is returned to the transaction destination account
- [x] B. Gas used by the transaction is credited to the beneficiary address in block header

- [ ] C. Unused gas is credited to the beneficiary address in block header
- [ ] D. Both A & B

▼ Solution

Correct is B.

> Gas refund and beneficiary: Any unused gas in a transaction (gasLimit minus gas used by the transaction) is refunded to the sender's account at the same gasPrice. Ether used to purchase gas used for the transaction is credited to the beneficiary address (specified in the block header), the address of an account typically under the control of the miner. This is the transaction "fees" paid to the miner.

from point 56 of Ethereum 101 - by Secureum

> "The high-level languages typically used for writing Ethereum smart contracts are"

— 20 of 32

- [ ] A. Go
- [ ] B. C++
- [x] C. Vyper
- [x] D. Solidity

▼ Solution

Correct is C, D.

> Solidity language continues to dominate smart contracts without much real competition (except Vyper perhaps).

from point 94 of Ethereum 101 - by Secureum

> "Ethereum nodes talk to each other via"

— 21 of 32

- [x] A. Peer-to-Peer network
- [ ] B. Client-Server network
- [ ] C. Satellite network
- [ ] D. None of the above

▼ Solution

Correct is A.

> Ethereum node/client: A node is a software application that implements the Ethereum specification and communicates over the peer-to-peer network with other Ethereum nodes.

from point 46 of Ethereum 101 - by Secureum

> “EVM is not a von Neumann architecture because”
>
> — 22 of 32

- [ ] A. It was co-founded by Vitalik Buterin, not John von Neumann
- [x] B. Program instructions are stored separately from data
- [ ] C. Program instructions are stored in a ROM not RAM
- [ ] D. It is quasi Turing complete

▼ Solution

Correct is B.

> EVM does not follow the standard von Neumann architecture. Rather than storing program code in generally accessible memory or storage, it is stored separately in a virtual ROM accessible only through a specialized instruction.

from point 64 of Ethereum 101 - by Secureum

> In computer science, a universal Turing machine (UTM) is a Turing machine that simulates an arbitrary Turing machine on arbitrary input. [...] This principle is considered to be the origin of the idea of a stored-program computer used by John von Neumann in 1946 for

> the "Electronic Computing Instrument" that now bears von Neumann's name: the von Neumann architecture.[1]

from Universal Turing machine - Wikipedia

> “User A sends transaction T1 from address A1 with gasPrice G1 and later transaction T2 from address A2 with gasPrice G2”
>
> — 23 of 32

- [ ] A. T1 will be always included in an earlier block than T2
- [ ] B. Inclusion/Ordering of these transactions depends only on gas prices G1 and G2
- [ ] C. Inclusion/Ordering of these transactions depends only on network congestion
- [x] D. Inclusion/Ordering of these transactions depends on miners

▼ Solution

Correct is D.

> Inclusion: Transaction inclusion is not guaranteed and depends on network congestion and gasPrice among other things. Miners determine inclusion.
> Order: Transaction order is not guaranteed and depends on network congestion and gasPrice among other things. Miners determine order.

from point 32 of Ethereum 101 - by Secureum

> “The types of accounts on Ethereum are”
>
> — 24 of 32

- [ ] A. All Accounts are the same
- [ ] B. Permissioned Accounts and Permissionless Accounts

- [x] C. Externally-Owned-Accounts and Contract Accounts
- [ ] D. User Accounts and Admin Accounts

▼ Solution

Correct is C.

> Ethereum has two different types of accounts:
> Externally Owned Accounts (EOAs) controlled by private keys
> Contract Accounts controlled by their contract code

from point 24 of Ethereum 101 - by Secureum

> "The number of decimals in Ether is"

— 25 of 32

- [ ] A. 0
- [ ] B. 1
- [x] C. 18
- [ ] D. Configurable

▼ Solution

Correct is C.

> Ethereum's currency unit is called ether or "ETH." Ether is subdivided into smaller units and the smallest unit is named wei. [...] and 10**18 wei is 1 Ether.

from point 17 of Ethereum 101 - by Secureum

> "The difference(s) between Bitcoin and Ethereum is/are"

— 26 of 32

- [x] A. The underlying tokens: Bitcoin vs Ether
- [x] B. Smart contract support

- [x] C. UTXO vs Accounts
- [ ] D. Nakamoto Consensus

▼ Solution

Correct is A, B, C.

See the Ethereum Whitepaper

> Consensus algorithm: Ethereum uses Bitcoin's consensus model, Nakamoto Consensus

from point 9 of Ethereum 101 - by Secureum

> "Security Audits for smart contracts are performed because"
>
> — 27 of 32

- [ ] A. They are required for listing DApp on the DApp store
- [ ] B. They are required for deployment on Ethereum
- [x] C. They help remove vulnerabilities and reduce risk
- [ ] D. They are required by exchanges to list tokens

▼ Solution

Correct is C.

> Audit-as-a-Silver-Bullet: Secure Software Development Lifecycle (SSDLC) processes for Web2 products have evolved over several decades to a point where they are expected to meet some minimum requirements of a combination of internal validation, external assessments (e.g. product/process audits, penetration testing) and certifications depending on the value of managed assets, anticipated risk, threat model and the market domain of products (e.g. financial sector has stricter regulatory compliance requirements).

from point 100 of Ethereum 101 - by Secureum

> “The number of modified Merkle-Patricia trees in an Ethereum block is”
>
> — 28 of 32

- [ ] A. One
- [ ] B. Three
- [x] C. Three plus number of contract accounts
- [ ] D. Three plus number of transactions included in the block

▼ Solution

Correct is C.

> Blocks contain block header, transactions and ommers’ block headers. Block header contains [...] `stateRoot`, `transactionsRoot` and `receiptsRoot` are 256-bit hashes of the root nodes of modified Merkle-Patricia trees.

from points 53, 54 of Ethereum 101 - by Secureum

> “EVM opcodes”
>
> — 29 of 32

- [ ] A. Are multi-byte instructions
- [x] B. Are single byte instructions
- [ ] C. Take operands in registers
- [x] D. Take operands on stack

▼ Solution

Correct is B, D.

> The code in Ethereum contracts is written in a low-level, stack-based bytecode language, referred to as "Ethereum virtual machine code" or "EVM code". The code consists of a series of bytes (hence called bytecode), where **each byte represents an operation**.

from points 58 of Ethereum 101 - by Secureum

> Most EVM instructions operate with the stack (stack-based architecture) and there are also stack-specific operations e.g. PUSH, POP, SWAP, DUP etc.

from points 60 of Ethereum 101 - by Secureum

> “Gas for EVM opcodes”
>
> — 30 of 32

- [ ] A. Is constant and the same for all opcodes
- [x] B. May be changed over time to prevent DoS attacks
- [ ] C. Depend on the gas price
- [ ] D. Depend on the miners

▼ Solution

Correct is B.

> Gas costs for different instructions are different depending on their computational/storage load on the client

from points 78 of Ethereum 101 - by Secureum

Correct is B.

> Tangerine Whistle — A hard fork to change the gas calculation for certain I/O-heavy operations and to clear the accumulated state from a denial-of-service (DoS) attack that exploited the low gas cost of those operations.
> Spurious Dragon — A hard fork to address more DoS attack vectors, and another state clearing. Also, a replay attack protection mechanism.

from "Ethereum’s Four Stages of Development" of Mastering Ethereum

> “Ethereum Virtual Machine is a”

— 31 of 32

- [ ] A. Register-based virtual machine
- [x] B. Stack-based virtual machine
- [ ] C. Heap-based virtual machine
- [ ] D. Stackless virtual machine

▼ Solution

Correct is B.

> The EVM is a simple stack-based architecture consisting of the stack, volatile memory, non-volatile storage with a word size of 256-bit (chosen to facilitate the Keccak256 hash scheme and elliptic-curve computations) and Calldata.

from points 59 of Ethereum 101 - by Secureum

> “Which of the following statement(s) is/are FALSE?”

— 32 of 32

- [ ] A. EVM can get the block number only of the current block
- [x] B. EVM can get the block hash only of the current block
- [x] C. EVM can get the account balance only of the current account
- [x] D. EVM can get the code hash of only the current account

▼ Solution

Correct is B, C, D.

> 0x31 BALANCE 1 1 Get balance of the given account
> 0x3f EXTCODEHASH 1 1 Get hash of an account’s code
> 0x40 BLOCKHASH 1 1 Get the hash of one of the **256 most recent complete blocks**
> 0x43 NUMBER 0 1 Get the block’s number

from points 70, 71 of Ethereum 101 - by Secureum

In Blockchain Tags Ethereum, Secureum Bootcamp

← Secureum Bootcamp Solidity 101 Q… CryptoHack CTF: Key Takeaways →