Quantstamp Contract Security Certificate

February 19th 2020 — Quantstamp Verified

Authereum

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

Executive Summary

Type	Protocol
Auditors	Alex Murashkin, Senior Software Engineer Ed Zulkoski, Senior Security Engineer Sung-Shine Lee, Research Engineer
Timeline	2020-01-21 through 2020-02-03
EVM	Istanbul
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Source Code	

Repository	Commit
authereum-contracts-audit	1d2bde8
authereum-contracts-audit	3b1552f

Total Issues	9 (2 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	0 (0 Resolved)
Low Risk Issues	3 (1 Resolved)
Informational Risk Issues	4 (1 Resolved)
Undetermined Risk Issues	2 (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	the issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.

Changelog

- 2020-01-31 - Initial report (commit `3b1552f`)
- 2020-02-03 - Final report (commit `1d2bde8`)
- 2020-02-19 - Report updated to mention the recent vulnerability disclosure report.

Overall Assessment

The scope of the audit was limited to the contracts located in two folders: `account` and `upgradeability`.
The code is overall well-written and documented. Quantstamp identified 9 issues: three low-severity, four informational, and two undetermined. No findings of medium or high severity were made.
The low-severity issues include missing input validation, transaction front-running, and a potential issue in the `createProxy` logic (which was addressed in `1d2bde8`).
The two findings: block timestamp manipulation and non-standard way of proxy implementation - were marked as undetermined due to lack of data to be able to assess the impact.
Update: the team has resolved the issues `QSP-2` and `QSP-5`, as well as made improvements to the documentation. For the remaining findings, the team provided an explanation, and they were marked as "Acknowledged".
Update 2: On February 17, the Authereum team received a disclosure report of a vulnerability that was not identified in the audit. The Authereum team quickly triaged the issue, deployed the patched version of the contract, and ensured that users retained ownership of their accounts. No funds were lost. The issue was fixed in the commit fdff18c. We acknowledge samczsun for the disclosure report.

Summary of Findings

ID	Description	Severity	Status
QSP-1	Missing parameter validation	Low	Acknowledged
QSP-2	Potential logic issue	Low	Resolved
QSP-3	Potential loss of relayer's funds when being front-run	Low	Acknowledged
QSP-4	Use of experimental features	Informational	Acknowledged
QSP-5	Incorrect return type	Informational	Resolved
QSP-6	"create2" dependency on initCode	Informational	Acknowledged
QSP-7	Centralization of Power	Informational	Acknowledged
QSP-8	Non-standard way of proxy implementation	Undetermined	Acknowledged
QSP-9	Block Timestamp Manipulation	Undetermined	Acknowledged

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Truffle
- Ganache
- SolidityCoverage
- Mythril
- Slither

Steps taken to run the tools:

1. Installed Truffle: `npm install -g truffle`
2. Installed Ganache: `npm install -g ganache-cli`
3. Installed the solidity-coverage tool (within the project's root directory): `npm install --save-dev solidity-coverage`
4. Ran the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
5. Installed the Mythril tool from Pypi: `pip3 install mythril`
6. Ran the Mythril tool on each contract: `myth -x path/to/contract`
7. Installed the Slither tool: `pip install slither-analyzer`
8. Run Slither from the project directory `slither .`

Assessment

Findings

QSP-1 Missing parameter validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `(multiple)`

Description:

- `AuthereumProxyFactory.sol`, `L29`: The constructor should validate that the parameters `_implementation` and `_authereumEnsManagerAddress` are non-zero
- `AuthereumProxyFactory.sol`, `L50`: `setAuthereumEnsManager(...)`should validate that the parameter `_authereumEnsManagerAddress` is non-zero
- `AuthrereumProxy.sol`, `L20` and `L67`: Consider adding a `require` statement to check `_logic` is not `null`
- `AuthereumEnsResolverProxy.sol`, `L22` and `L67`: Consider adding a `require` statement to check `_logic` is not `null`
- `AuthereumProxy.sol`, `L90`: `_salt` should be validated to be non-empty
- `AuthereumProxy.sol`, `L110` (commit `3b1552f`): `_label` is used but not validated

Recommendation: It is recommended to validate inputs even if a method is owner-only, because invalid inputs can be passed for a variety of reasons (e.g., malfunctioning of a script that calls the contract).
Update: the team has acknowledged the finding and provided the following explanation: "All suggested validation is done off-chain, both before and after the relevant transaction. Some of the non-validation on-chain was done for efficiency purposes. No updates will be made to the contracts at this time."

QSP-2 Potential logic issue

Severity: *Low Risk*

Status: Resolved

File(s) affected: `AuthereumProxyFactory.sol`

Description: `L103`: `if(_initData.length > 0)` - likely, needs to be `_initData[i].length`

Recommendation: Fixing the code as suggested above.

QSP-3 Potential loss of relayer's funds when being front-run

Severity: *Low Risk*

Status: Acknowledged

Description: In case of multiple relayers, there is a possibility of a relayer losing their funds when being out-run by other relayers.

Exploit Scenario:

1. Relayer A estimates a transaction to be relayed and concludes that it is non-detrimental to them
2. Relayer A sends a transaction, it becomes pending
3. Relayer B sees the transaction broadcasted to the network and creates its own transaction
4. Relayer B submits the transaction with a gas price that is higher than the original transaction
5. Relayer B's transaction gets mined, and Relayer B gets the token reward
6. Relayer A's transaction fails because Relayer B already relayed a similar transaction and got rewarded
7. Relayer A loses some funds due to gas spending.

Recommendation: While there is no immediate mitigation strategy, we recommend documenting such a risk and communicating it to potential relayers.
Update: the team has acknowledged the finding and provided the following note: "There are known griefing vectors on the relayer layer. Many of these griefing vectors are handled by the logic in our relayers and their interactions with each other and with other relayers. No updates will be made to the contracts at this time."

QSP-4 Use of experimental features

Severity: *Informational*

Status: Acknowledged

File(s) affected: `(multiple)`

Description: The project is using `pragma experimental ABIEncoderV2`, which enables an experimental version of the ABI decoder. Experimental features may contain bugs, such as, this.

Recommendation: While we are not aware of any immediate issues, use of such features could lead to risks. We recommend staying up-to-date with regards to any new ABIEncoderV2-related issues and being able to address them in a timely manner.
Update: the team has acknowledged the finding and provided the following explanation: "We chose to use Solidity version 0.5.16 for our contracts due to all of the bug fixes of experimental features in prior versions. This version is the final version before 0.6.0, which marks ABIEncoderV2 as no longer experimental."

QSP-5 Incorrect return type

Severity: *Informational*

Status: Resolved

Description: The return type of `executeMultipleMetaTransactions()` does not appear correct, since the underlying function call only returns `bytes[]`

Recommendation:

1. Fixing the return type
2. Making sure the return values are propagated to the caller by adding the `return` statement.

QSP-6 "create2" dependency on initCode

Severity: *Informational*

Status: Acknowledged

File(s) affected: `AuthereumProxyFactory.sol`

Description: `create2` allows users to calculate their wallet address before deploying the actual smart contract. Thus it is possible that they receive funds before the wallet is being deployed. Since `create2` address calculation also depends on the `initCode`, if the `initCode` is changed, a user cannot deploy the wallet contract at the previous address even when the same `salt` is provided. Also, since `create2` address calculation also depends on the `msg.sender` (which in this case would be the address of the `proxyFactory`), it is not feasible for the user to deploy the contract by themselves.

Recommendation: After checking with the team, it was mentioned that `create2` is used as purely deployment method, and there is no plan to use addresses before the contract is deployed. It is suggested to clearly communicate to users that no funds should be sent to the address before contract deployment.
Update: the team has acknowledged the finding and noted the following: "This was done by design. We will more clearly communicate to users that no funds should be sent to the address before contract deployment."

QSP-7 Centralization of Power

Severity: *Informational*

Status: Acknowledged

File(s) affected: `AuthereumEnsResolverProxy.sol`

Description: Smart contracts will often have `owner` variables to designate the person with special privileges to make modifications to the smart contract. In case of `AuthereumEnsResolverProxy`, users need to trust the owner to change the implementation responsibly. Changing to an incorrect implementation may break important functionality to users.

Recommendation: This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.
Update: the team has acknowledged the finding and noted the following: "This was done by design. The ownership of certain functions will be set in such a way that users are made aware of any changes in advance of them being made."

QSP-8 Non-standard way of proxy implementation

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `AuthereumProxy.sol`, `AuthereumEnsResolverProxy.sol`

Description: `AuthereumProxy.sol`, `L39` and `AuthereumEnsResolverProxy`, `L45`: Typical proxy implementations use the free memory pointer `0x40` instead of `0` to store the calldata, for instance, here. The documentation states the suggested way of approaching to memory management:
`Solidity manages memory in a very simple way: There is a “free memory pointer” at position 0x40 in memory. If you want to allocate memory, just use the memory from that point on and update the pointer accordingly.`
The benefit of using 0 instead remains unclear.

Recommendation: It is recommended to use the industry standard for proxy implementation, unless there is a clear benefit of doing otherwise. Deviations from the standard implementation could potentially lead to unknown issues in the future.
Update: the team has acknowledged the finding and stated the following: "This was done by design because of the simplicity of the proxy contract. Per the contract comments, `Copy msg.data. We take full control of memory in this inline assembly block because it will not return to Solidity code. We overwrite the Solidity scratch pad at memory position 0.`"

QSP-9 Block Timestamp Manipulation

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `LoginKeyMetaTxAccount.sol`

Description: Projects may rely on block timestamps for various purposes. However, it's important to realize that miners individually set the timestamp of a block, and attackers may be able to manipulate timestamps for their own purposes. If a smart contract relies on a timestamp, it must take this into account. `LoginKeyMetaTxAccount.sol`, `L89` uses `now` for comparison. Depending on the precision required, this may or may not be an issue.

Recommendation: Clarifying the precision requirements for timestamps.
Update: the team has clarified that the timestamp use in this context is not granular enough to be manipulated.

Automated Analyses

Mythril

Mythril was unable to run for the given project.

Slither

Slither was unable to run for the given project.

Code Documentation

The code appears to be well-documented.

Some minor issues:

- `BaseMetaTxAccount.sol`, `L140` and `L147`: typo: `propogate` - **Fixed**
- `README.md`, `L62`: typo: "Verifiy Signatures" - **Fixed**
- `README.md`, multiple lines: typo: "complie" - **Fixed**
- `README.md`, `L107`: "runnning" - **Fixed**
- `README.md`, `L75`: "Authereum's contracts are upgradeable. Each Authereum user owns a proxy (`AuthereumProxyFactory.sol`)" what is meant is likely “AuthereumProxy” - **Fixed**
- `AccountStateV1.sol`, `L7`: typo: "abscraction" - **Fixed**
- `LoginKeyMetaTxAccount.sol`, `L100`: a typo: `_transactionMessgeHashSignature`

Adherence to Best Practices

- `BaseAccount.sol`, `L62`: `authKeys[_authKey] == false` could be `!authKeys[_authKey]`
- `BaseAccount.sol`, `L20`: `SafeMath` is not needed
- `AuthMetaTxAccount.sol`, `L19`: the comment "_feeTokenRate Rate of the token (in tokenGasPrice/ethGasPrice) used to pay a fee" is not entirely accurate, since in the case of ETH refunds, this parameter is not used by the underlying `_issueRefund()` function
- `package.json`: package versions are not fixed, and `package-lock.json` is not in the source control. It is recommended to use fixed versions to avoid pulling unintentional breaking changes upon next re-install

Update: the team stated that "No changes will be made here, as some of these were done by design and others are not critical for deployment."

Test Results

Test Suite Results

All tests were passing (with the exception of those that aren't enabled).

```
  Contract: AccountUpgradeability
    upgradeToAndCall
      Happy Path
         ✓ Should upgrade a proxy's logic address (w/o init) (289ms)
         ✓ Should upgrade a proxy's logic address (w/ init) (324ms)
      Non-Happy Path
         ✓ Should not allow an arbitrary account to call this function (75ms)
         ✓ Should not allow a proxy's to upgrade w/ init in an incorrect order (208ms)
         ✓ Should not allow a proxy's to upgrade w/ init in a non-contract address as the implementation address
(190ms)

  Contract: AuthereumAccount
    authereumVersion
      Happy path
         ✓ Should return the Authereum contract version
    initialize
      Happy path
         ✓ Should initialize an upgradable contract (267ms)
         ✓ Should initialize an upgradable contract and upgrade the contract (285ms)
      Non-Happy path
         ✓ Should not initialize an upgradable contract because a _label has already been used (164ms)
         ✓ Should not allow Authereum to upgrade a proxy for a user (54ms)

  Contract: AuthKeyMetaTxAccount
    executeMultipleAuthKeyMetaTransactions
      Happy Path
         ✓ Should successfully execute an auth key meta transaction (189ms)
         ✓ Should successfully verify and sign two transactions (378ms)
         ✓ Should successfully verify and sign two transactions (batched) (208ms)
         ✓ Should successfully verify and sign two transactions (batched) and increment the contract nonce by 2
(217ms)
         ✓ Should successfully execute an auth key meta transaction and pay fees in tokens (343ms)
         ✓ Should successfully verify and sign two transactions (batched) (403ms)
         ✓ Should successfully execute an auth key meta transaction and pay fees in tokens that have a non-
standard decimals (decimals should affect the rate and nothing else) (346ms)
         ✓ Should successfully execute an auth key meta transaction when the relayer sends a higher gasPrice than
expected (189ms)
         ✓ Should successfully execute an auth key meta transaction and not pay fees because it is a self upgrade
(179ms)
         ✓ Should send two transactions to self and not pay any fees (relayer pays all fees) (264ms)
      Non-Happy Path
        Bad Parameters
           ✓ Should revert due to the relayer sending too small of a gasPrice with the transaction (77ms)
           ✓ Should emit a CallFailed event due to failed transaction because of bad data (145ms)
           ✓ Should increment the contract nonce by 2 even though the second of 2 atomic transactions failed (and
should rewind the first) (226ms)
           ✓ Should revert due to not enough funds being in the contract to send the transaction (282ms)
           ✓ Should revert due to not enough funds being in the contract to send the refund (187ms)
         - Should revert due to too low of a gasLimit sent with the transaction
           ✓ Should fail to send a transaction due to a bad signed message (161ms)
           ✓ Should refund the relayer even though one (of two) transactions is sent to self (262ms)
           ✓ Should not refund the relayer for InvalidAuthkey (194ms)
           ✓ Should emit a CallFailed event due to incorrect transaction params (address in uint256) (133ms)
           ✓ Should revert due to incorrect transaction params (bytes in the addr param) (164ms)
           ✓ Should throw and cost the relayer if the account does not send a large enough gasLimit (ETH) (201ms)
           ✓ Should throw and cost the relayer if the account does not send a large enough gasLimit (tokens)
(282ms)
           ✓ Should revert due to the fact that the relayer used a different token address than expected (326ms)
           ✓ Should revert due to the fact that the relayer used a different token rate than expected (285ms)

  Contract: BaseAccount
    fallback
       ✓ Should allow anyone to send funds to the contract (137ms)
       ✓ Should use exactly 21084/21084 gas (depending on the fork) on a transaction with no data (40ms)
       ✓ Should use exactly 22654/23276 gas (depending on the fork) on a transaction with data (48ms)
    getChainId
       ✓ Should return a chain ID of 1
    addAuthKey
      Happy Path
         ✓ Should add an authKey (105ms)
         ✓ Should add two authKeys (199ms)
         ✓ Should add an authKey through executeMultipleAuthKeyMetaTransactions (196ms)
      Non-happy Path
         ✓ Should not add the same authKey twice (61ms)
         ✓ Should not allow a random address to add an auth key (61ms)
         ✓ Should not allow a loginKey to add an authKey through executeMultipleLoginKeyMetaTransactions (126ms)
    removeAuthKey
      Happy Path
         ✓ Should remove an authKey (196ms)
         ✓ Should add two authKeys and then remove two authKeys (380ms)
         ✓ Should add two authKeys and then remove two authKeys in reverse order (371ms)
         ✓ Should add an authKey and then remove the original authKey (202ms)
         ✓ Should remove an authKey through executeMultipleAuthKeyMetaTransactions (255ms)
      Non-Happy Path
         ✓ Should not remove an authKey that was never a added (70ms)
         ✓ Should not allow a user to remove all authKeys (63ms)
         ✓ Should not allow a random address to remove an auth key (118ms)
         ✓ Should not allow a loginKey to remove an authKey through executeMultipleLoginKeyMetaTransactions
(165ms)

  Contract: BaseMetaTxAccount
    executeMultipleMetaTransactions
      Happy Path
         ✓ Should execute a single transaction from an auth key (no refund) (125ms)
         ✓ Should execute a two transaction (batched) (no refund) (158ms)
      Non-Happy Path
         ✓ Should revert if the transaction fails (137ms)
         ✓ Should revert if a random address tries to call it (106ms)

  Contract: ERC1271Account
```

```
    isValidSignature
      Happy Path
         ✓ Should return the magic value for a login key signature (298ms)
         ✓ Should return the magic value for an auth key signature (231ms)
         ✓ Should return INVALID_SIG for isValidLoginKeySignature() due to a signature of length > 130 but bad
data (209ms)
      Non-Happy Path
         ✓ Should not return the magic value for a login key signature due to bad message (223ms)
         ✓ Should not return the magic value for an auth key signature due to bad message (159ms)
         ✓ Should revert isValidSignature() due to a signature of length < 65 (71ms)
         ✓ Should revert isValidSignature() due to a signature of length > 65 and < 130 (68ms)
         ✓ Should revert isValidAuthKeySignature() due to a signature of length != 65
         ✓ Should revert isValidLoginKeySignature() due to a signature of length < 130

  Contract: LoginKeyMetaTxAccount
    executeMultipleLoginKeyMetaTransactions
      Happy Path
         ✓ Should successfully execute a login key meta transaction (190ms)
         ✓ Should successfully verify and sign two transactions (366ms)
         ✓ Should successfully verify and sign two transactions (batched) (220ms)
         ✓ Should successfully verify and sign two transactions (batched) and increment the contract nonce by 2
(223ms)
         ✓ Should successfully execute a login key meta transaction and pay fees in tokens (403ms)
         ✓ Should successfully verify and sign two transactions (batched) and pay fees in tokens (438ms)
         ✓ Should successfully execute a login key meta transaction and pay fees in tokens that have a non-
standard decimals (decimals should affect the rate and nothing else) (381ms)
         ✓ Should successfully execute a login key meta transaction when the relayer sends a higher gasPrice than
expected (224ms)
        - Should successfully execute a login key meta whose value is equivalent to the daily limit
        - Should successfully execute a login key meta whose value is equivalent to the daily limit, add to the
limit, and do the same thing
      Non-Happy Path
        Bad Parameters
           ✓ Should revert due to the relayer sending too small of a gasPrice with the transaction (90ms)
          - Should throw due to surpassing the daily limit
          - Should emit OverDailyLimit due to surpassing the daily limit after 2 transactions
           ✓ Should fail to send a transaction due to a failed transaction because of bad data (137ms)
           ✓ Should increment the contract nonce by 2 even though the second of 2 atomic transactions failed (and
should rewind the first) (234ms)
           ✓ Should revert due to not enough funds being in the contract to send the transaction (297ms)
           ✓ Should revert due to not enough funds being in the contract to send the refund (196ms)
          - Should revert due to too low of a gasLimit sent with the transaction
           ✓ Should fail to fail to send a transaction due to a bad signed message (230ms)
           ✓ Should not refund the relayer for InvalidTransactionDataSigner (277ms)
          - Should not refund the relayer for OverDailyLimit
           ✓ Should throw and cost the relayer if the account does not send a large enough gasLimit (ETH) (212ms)
           ✓ Should throw and cost the relayer if the account does not send a large enough gasLimit (tokens)
(325ms)
           ✓ Should throw because the login key is trying to upgrade the proxy (157ms)
           ✓ Should emit a CallFailed event due to incorrect transaction params (address in uint256) (152ms)
           ✓ Should revert due to incorrect transaction params (bytes in the addr param) (83ms)
           ✓ Should revert due to the fact that the relayer used a different token address than expected (403ms)
           ✓ Should revert due to the fact that the relayer used a different token rate than expected (269ms)
        Invalid Permissions
           ✓ Should revert if login key is expired (153ms)

Contract: Timelock
  Constructor
     ✓ Should return the correct timelock time
     ✓ Should return the correct timelockExpire time
  getUnlockTime
     ✓ Should return the correct timelock time for an uninitialized piece of data
     ✓ Should return the correct timelock time for an initialized piece of data (95ms)
  getUnlockExpireTime
     ✓ Should return the correct expire timelock time for an uninitialized piece of data
     ✓ Should return the correct expire timelock time for an initialized piece of data (89ms)
  getRemainingUnlockTime
     ✓ Should return the entire unlock time (79ms)
     ✓ Should return the half unlock time (106ms)
     ✓ Should return 0 for the unlock time since it has already passed (90ms)
     ✓ Should return 0 for the unlock time since it has not yet been initiated
  getRemainingUnlockExpireTime
     ✓ Should return the entire unlock expire time (81ms)
     ✓ Should return the unlock expire time after a month (89ms)
     ✓ Should return 0 for the unlock expire time since it has already passed (90ms)
     ✓ Should return 0 for the unlock expire time since it has not yet been initiated
  getCurrentChangeState
     ✓ Should return the correct changeState for an uninitialized piece of data
     ✓ Should return the correct changeState for a pending piece of data (79ms)
     ✓ Should return the correct changeState for a changeable piece of data (87ms)
     ✓ Should return the correct changeState for an expired piece of data (97ms)
  setTimelock
     ✓ Should set the timelock from a change process (153ms)
     ✓ Should not let the owner call this funciton directly (40ms)
     ✓ Should not let anyone call this funciton directly (49ms)
  setTimelockExpire
     ✓ Should set the timelock from a change process (145ms)
     ✓ Should not let the owner call this funciton directly (48ms)
     ✓ Should not let anyone call this funciton directly (50ms)
  initiateChange
     ✓ Should set the state of a change to pending, set the unlock time to a month from now, set the unlock
expire time to one month + one week from now, and trigger an event (144ms)
     ✓ Should not allow an non-uninitialized data and address combination to be initialized (108ms)
  executeChange
     ✓ Should execute the change, reset the state of the data and address pair to uninitialized, and emit an
event (183ms)
     ✓ Should execute the change that sets a new timelock and sends 0.1 ETH to the receiving (197ms)
     ✓ Should not execute the change due to the change still being in a pending state (117ms)
     ✓ Should not execute the change due to the change already being made (170ms)
  cancelChange
     ✓ Should cancel a pending change (170ms)
     ✓ Should cancel a changeable change (247ms)
     ✓ Should cancel an expired change (130ms)

Contract: Managed
  addManager
    Happy Path
       ✓ Should add a new manager (64ms)
       ✓ Should do nothing if the same address is set as an owner (79ms)
    Non-Happy Path
       ✓ Should not allow 0 to be a manager (43ms)
       ✓ Should exclusively allow the owner to set a manager (42ms)
  revokeManager
    Happy Path
       ✓ Should remove a manager (111ms)
    Non-Happy Path
       ✓ Should not remove a manager if said manager is not already set (42ms)
       ✓ Should not remove a manager if the function is not called by the owner (46ms)

Contract: Owned
  isOwner
    Happy Path
       ✓ Should return true if the owner is passed in
       ✓ Should return false if the owner is not passed in
  changeOwner
    Happy Path
```

```
        ✓ Should allow the owner to change the owner (63ms)
    Non-Happy Path
        ✓ Should not allow a non-owner to change the owner (44ms)
        ✓ Should not allow the owner to be set to 0 (45ms)

Contract: AuthereumEnsManager
  Sanity Checks
    Happy Path
        ✓ Should set the Authereum ENS Manager as the owner of authereum.eth
        ✓ Should return the Authereum resolver address for a subdomain
        ✓ Should return address(0) for the authereumDotEthNode resolver (47ms)
        ✓ Should return address(0) for an unclaimed subdomain resolver
  getEnsRegistry
    Happy Path
        ✓ Should return the current registry
  getEnsReverseRegistrar
    Happy Path
        ✓ Should return the ENS Reverse Registrar
  changeRootnodeOwner
    Happy Path
        ✓ Should update the owner of the rootNode from the manager address to a new manager address (174ms)
    Non-Happy Path
        ✓ Should not allow an arbitrary actor to update the rootnode owner (89ms)
        ✓ Should not allow the rootnode owner to be set to 0 (54ms)
  changeRootnodeResolver
    Happy Path
        ✓ Should update the resolver of the rootNode from the manager address to a new resolver (162ms)
    Non-Happy Path
        ✓ Should not allow an arbitrary actor to update the rootnode resolver (51ms)
        ✓ Should not allow the rootnode resolver to be set to 0 (69ms)
  changeRootnodeTTL
    Happy Path
        ✓ Should update the TTL of the rootNode from the 0 to 1 (104ms)
    Non-Happy Path
        ✓ Should not allow an arbitrary actor to update the rootnode ttl (52ms)
  changeRootnodeText
    Happy Path
        ✓ Should update the text record of the rootNode from the 0 to the default key and value (142ms)
    Non-Happy Path
        ✓ Should not allow an arbitrary actor to update the rootnode text (57ms)
  changeRootnodeContenthash
    Happy Path
        ✓ Should update the contenthash of the rootNode from the null to the default contenthash (177ms)
    Non-Happy Path
        ✓ Should not allow an arbitrary actor to update the rootnode contenthash (54ms)
  changeAuthereumFactoryAddress
    Happy Path
        ✓ Should update the owner of the Authererеum Factory address to a new Authereum Factory address (48ms)
    Non-Happy Path
        ✓ Should not allow an arbitrary actor to update the Authereum Factory address (58ms)
        ✓ Should not allow the Authereum Factory address to be set to 0 (53ms)
  changeAuthereumEnsResolver
    Happy Path
        ✓ Should update the owner of the ENS Resolver address to a new ENS Resolver address (151ms)
    Non-Happy Path
        ✓ Should not allow an arbitrary actor to update the ENS Resolver address (40ms)
        ✓ Should not allow the new ENS resolver address to be set to 0 (116ms)
  Register
    Happy Path
        ✓ Should let a user register test.authereum.eth (104ms)
        ✓ Should let a user register test.authereum.eth and another user to register testtwo.authereum.eth
(216ms)
        ✓ Should let a user register test.authereum.eth and another user register Test.authereum.eth, but
ownership will not change because namehash normalizes the names to be the same case (198ms)
    Non-Happy Path
        ✓ Should not allow a domain name to be registered more than once (94ms)
        ✓ Should not allow an non-authereumProxyFactory address to register an account (102ms)
  isAvailable
    Happy Path
        ✓ Should return true if a subnode is available
        ✓ Should return false if a subnode is not available
  End to End
    Happy Path
        ✓ Should update to a new manager and retain all qualities as before the upgrade (345ms)
        ✓ Should update to a new manager and retain all qualities as before the upgrade, including users (509ms)

Contract: AuthereumEnsResolver
  setAddr
    Happy Path
        ✓ Should allow a manager (multisig) to setAddr (77ms)
    Non-Happy Path
        ✓ Should not allow an owner to change their addr (68ms)
        ✓ Should not allow an arbitrary user to change an addr (53ms)
  setName
    Happy Path
        ✓ Should allow a manager (multisig) to setName (89ms)
    Non-Happy Path
        ✓ Should not allow an owner to change their name (51ms)
        ✓ Should not allow an arbitrary user to change an name (50ms)
  setText
    Happy Path
        ✓ Should allow a manager (multisig) to setText (89ms)
    Non-Happy Path
        ✓ Should not allow an owner to change their text (54ms)
        ✓ Should not allow an arbitrary user to change a text (56ms)
  setContenthash
    Happy Path
        ✓ Should allow a manager (multisig) to setContenthash (93ms)
    Non-Happy Path
        ✓ Should not allow an owner to change their contenthash (57ms)
        ✓ Should not allow an arbitrary user to change a contenthash (61ms)
  supportsInterface
    Happy Path
        ✓ Should return true for meta interface
        ✓ Should return true for addr interface (56ms)
        ✓ Should return true for name interface (47ms)
        ✓ Should return true for text interface
        ✓ Should return true for contenthash
  End to end
    Happy Path
        ✓ Should allow a manager (multisig) to setAddr, remove the manager, add another manager, and add setAddr
for another account (280ms)

Contract: AuthereumEnsResolverProxy
  fallback
    Non-Happy Path
      - Should allow an arbitrary person to call the fallback but not change any state on behalf of the proxy
owner
  implementation
    Happy Path
        ✓ Should confirm the implementation address after the creation of a proxy

Contract: AuthereumProxy
  fallback
    Non-Happy Path
      - Should allow an arbitrary person to call the fallback but not change any state on behalf of the proxy
```

```
owner
    implementation
      Happy Path
         ✓ Should confirm the implementation address after the creation of a proxy

  Contract: AuthereumProxyFactory
    setInitCode
      Happy Path
         ✓ Should correctly set the new initCode (132ms)
    setAuthereumEnsManager
      Happy Path
         ✓ Should correctly set the authereumEnsManager (67ms)
    getInitCode
      Happy Path
         ✓ Should correctly get the initCode (66ms)
    getAuthereumEnsManager
      Happy Path
         ✓ Should correctly get the authereumEnsManager
    createProxy
      Happy Path
         ✓ Should create a proxy based on the creationCode (no init data) (193ms)
         ✓ Should create a proxy based on the creationCode (1 init data) (264ms)
         ✓ Should create a proxy based on the creationCode (multiple init data) (703ms)
         ✓ Should create a proxy based on the creationCode (1 init data, 1 non-init data) (683ms)
      Non-Happy Path
         ✓ Should fail to create a proxy due to a reused label (281ms)
         ✓ Should fail to create a proxy based on bad init data (243ms)


  197 passing (1m)
  9 pending
```

Code Coverage

We encountered an issue when running Solidity-coverage.
Unfortunately, we cannot accurately measure test coverage for the given files.

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
9c90c49b0b44c442810c993208207984df31eb5bdf1bc3ed7b8469c3e280e304
./contracts/upgradeability/AuthereumEnsResolverProxy.sol

5942396b52c733f4928847deef9e653578f15fdb67078d4f866b52b76106f194 ./contracts/upgradeability/AuthereumProxy.sol

57fd1af97170a7a0e6c0fef02d5d53351657592d55c6f8b89b001e4f0cb5c769
./contracts/upgradeability/AuthereumProxyFactory.sol

c7d702f33936439c3397472b90be3a9c8a1d1b370f7d91415aaea8a99c01da9b ./contracts/account/AccountUpgradeability.sol

f6c02eb30102efa0cd58b0b902e3da539c8161c06539c56a9347f090345b20fc ./contracts/account/AuthereumAccount.sol

35882049f54ce68b09a39ef9c9345ac7545721b594ce9bdb0aae27735b9f98b7 ./contracts/account/AuthKeyMetaTxAccount.sol

41e8e98f2d0f1e9959bf769519c1592c9de8f4f9dbd34a3557343b881434c023 ./contracts/account/BaseAccount.sol

4665d4e383ce6f8067a49046246f87f5154d40eb80d00c5f12a0c8f3185f5824 ./contracts/account/BaseMetaTxAccount.sol

cbd13821d22fc0fb775b3a3cdc425cc1d883575a61334dbbb8b402d66983eef6 ./contracts/account/ERC1271Account.sol

f3bc12ad0f811cfe1ea24e5140f5eb7247b1ec0136bbd19b7102bf78494ca529 ./contracts/account/LoginKeyMetaTxAccount.sol

ea1045a06487570e58fd01736cc267849a9b9235c131d11d70a08e3eb2a62a45 ./contracts/account/TokenReceiverHooks.sol

173377cbc39227d026e4b0cb14f2eb3cade428f243bc63524c79c5297b50e3e3 ./contracts/account/state/AccountState.sol

89ff4c8a1cf15bb335b3af8e1fde8dd66e237868f3a2a7a7b521a91e546ec7a5 ./contracts/account/state/AccountStateV1.sol

77490230ff0184e7b1fda173f0d85afd5449906f7edfff68f4d743f66e0fd364
./contracts/account/initializer/AccountInitialize.sol

d45db84fcad6cb63214e249bcace67a4e9f0d6dfaa906a5439cba30efbe7550b
./contracts/account/initializer/AccountInitializeV1.sol

9807154c37bc24b4b9fb6da631e60b2d2782ff46fda64dd2fe99afc3750fe81a ./contracts/account/event/AccountEvents.sol
```

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure smart contracts at scale using computer-aided reasoning tools, with a mission to help boost adoption of this exponentially growing technology.

Quantstamp's team boasts decades of combined experience in formal verification, static analysis, and software verification. Collectively, our individuals have over 500 Google scholar citations and numerous published papers. In its mission to proliferate development and adoption of blockchain applications, Quantstamp is also developing a new protocol for smart contract verification to help smart contract developers and projects worldwide to perform cost-effective smart contract security audits.

To date, Quantstamp has helped to secure hundreds of millions of dollars of transaction value in smart contracts and has assisted dozens of blockchain projects globally with its white glove security auditing services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Finally, Quantstamp's dedication to research and development in the form of collaborations with leading academic institutions such as National University of Singapore and MIT (Massachusetts Institute of Technology) reflects Quantstamp's commitment to enable world-class smart contract innovation.