Quantstamp Security Assessment Certificate

March 25th 2021 — Quantstamp Verified

# Illuvium ERC20

This security assessment was prepared by Quantstamp, the leader in blockchain security.

## Executive Summary

| | |
|---|---|
| Type | ERC20 Token |
| Auditors | Kacper Bąk, Senior Research Engineer<br>Sebastian Banescu, Senior Research Engineer |
| Timeline | 2021-03-10 through 2021-03-19 |
| EVM | Muir Glacier |
| Languages | Solidity, Javascript |
| Methods | Architecture Review, Unit Testing, Computer-Aided Verification, Manual Review |
| Specification | None |
| Documentation Quality | High |
| Test Quality | High |

Source Code

| Repository | Commit |
|---|---|
| illuvium-contracts | c9e0327 |

Goals

- Does the contract conform to ERC20?
- Is voting power proportional to the tokens held by account?

| | |
|---|---|
| Total Issues | 6 (4 Resolved) |
| High Risk Issues | 0 (0 Resolved) |
| Medium Risk Issues | 0 (0 Resolved) |
| Low Risk Issues | 4 (3 Resolved) |
| Informational Risk Issues | 2 (1 Resolved) |
| Undetermined Risk Issues | 0 (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. |
| Mitigated | Implemented actions to minimize the impact or likelihood of the risk. |

## Summary of Findings

We have found only a few low and informational severity issues within the reviewed code. The scope of the review has been limited to the following contracts: `IlluviumERC20.sol` and `AccessControl.sol`. Overall, the code follows best practices and is well documented and well tested. We recommend, however, addressing all the issues before deploying the contracts.
**Update:** as of commit `903a787` two issues have been fixed, one issue has been mitigated and the remaining two issues have been acknowledged. We did not evaluate any other changes between this and the initially reviewed commit.

| ID | Description | Severity | Status |
|---|---|---|---|
| QSP-1 | Privileged Roles and Ownership | Low | Acknowledged |
| QSP-2 | Wrong underflow check | Low | Fixed |
| QSP-3 | `FeaturesUpdated` event is never used | Low | Fixed |
| QSP-4 | Off-by-one error | Low | Fixed |
| QSP-5 | Allowance Double-Spend Exploit | Informational | Mitigated |
| QSP-6 | Unsafe way to check if address is contract or EOA | Informational | 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.

## Findings

### QSP-1 Privileged Roles and Ownership

Severity: *Low Risk*

**Status:** Acknowledged

**File(s) affected:** `token/IlluviumERC20.sol`

**Description:** Smart contracts will often have `owner` variables (or other privileged roles) to designate the person with special privileges to make modifications to the smart contract. The following privileged roles and capabilities exist in the system:

1. `ROLE_TOKEN_CREATOR` can mint arbitrary amounts of ILV to any account at any time.
2. `ROLE_TOKEN_DESTROYER` can burn arbitrary amount of ILV from any account at any time.

**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 informed us that mint and burn roles are required by design, but they are to be revoked during the token lifecycle.

## QSP-2 Wrong underflow check

Severity: *Low Risk*

**Status:** Fixed

**File(s) affected:** `token/IlluviumERC20.sol`

**Description:** In L742-745, the checks are wrong. The check in L742 tests for overflow (instead of underflow), whereas the check in L745 allows to decrease by 0 (which contradicts the documentation).

**Recommendation:** We recommend replacing both checks with the following: `currentVal > _value`. Although the incorrect checks may confuse users of the function `decreaseAllowance()`, we marked the issue as low severity since the allowance can always be set to 0 using `approve()`.

## QSP-3 `FeaturesUpdated` event is never used

Severity: *Low Risk*

**Status:** Fixed

**File(s) affected:** `utils/AccessControl.sol`

**Description:** `FeaturesUpdated` event defined in `AccessControl.sol` contract is never used. Instead, the `RoleUpdated` event is emitted when features are updated.

**Recommendation:** We recommend removing the unused event.

## QSP-4 Off-by-one error

Severity: *Low Risk*

**Status:** Fixed

**File(s) affected:** `token/IlluviumERC20.sol`

**Description:** Signature expiration should occur at the block with timestamp specified by the `_exp` input parameter of the `delegateWithSig()` function. This means that at `block.timestamp == _exp` the signature should already be expired. However, the `require` statement on L1014 only throws if `block.timestamp > _exp`.

**Recommendation:** Change the condition in the `require` statement on L1014 to `block.timestamp < _exp`.

## QSP-5 Allowance Double-Spend Exploit

Severity: *Informational*

**Status:** Mitigated

**File(s) affected:** `token/IlluviumERC20.sol`

**Description:** As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

**Exploit Scenario:**

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

**Recommendation:** The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

## QSP-6 Unsafe way to check if address is contract or EOA

Severity: *Informational*

**Status:** Acknowledged

**File(s) affected:** `token/IlluviumERC20.sol`

**Description:** The way in which `safeTransfer()` checks whether the `_to` address is a contract or an EOA is not always safe, because it uses the `extcodesize` instruction, which returns 0 indicating an EOA, when it is called from the `constructor()` of the `_to` address. This would effectively allow transferring ILV to a contract which is not a valid `ERC20Receiver` via the `safeTransferFrom()` function.

**Recommendation:** We have no recommendation but wanted to indicate the limitations of this check.

**Update:** The team confirmed that the issue does not affect token related business logic and token functional requirements.

# Code Documentation

The code is well documented.

# Adherence to Best Practices

The code follows best practices.

# Test Results

### Test Suite Results

```
Contract: IlluviumERC20 AccessControl (ACL) tests
  ACL Core
    when performed by ACCESS_MANAGER
      when ACCESS_MANAGER has full set of permissions
        what you set
          ✓ is what you get (42ms)
        what you remove
          ✓ is what gets removed
      when ACCESS_MANAGER doesn't have any permissions
        what you get, independently of what you set
          ✓ is always zero (47ms)
        what you get, independently of what you remove
          ✓ is always what you had
      when ACCESS_MANAGER has some permissions
        what you get
          ✓ is an intersection of what you set and what you have
        what you remove
          ✓ is an intersection of what you tried to remove and what you have
      ACCESS_MANAGER updates itself
        ✓ and degrades to zero (656ms)
      when ACCESS_MANAGER grants ACCESS_MANAGER permission
        ✓ operator becomes an ACCESS_MANAGER (55ms)
      when ACCESS_MANAGER revokes ACCESS_MANAGER permission from itself
        ✓ operator becomes an ACCESS_MANAGER (54ms)
    otherwise (no ACCESS_MANAGER permission)
      ✓ updateFeatures reverts (52ms)
      ✓ updateRole reverts (48ms)
  ACL Illuvium ERC20
    after spender's approval
      when FEATURE_TRANSFERS_ON_BEHALF is enabled
        ✓ transfer on behalf succeeds (60ms)
      when FEATURE_TRANSFERS_ON_BEHALF is disabled
        ✓ transfer on behalf reverts (56ms)
      when FEATURE_BURNS_ON_BEHALF is enabled
        ✓ burn on behalf succeeds (60ms)
      when FEATURE_BURNS_ON_BEHALF is disabled
        ✓ burn on behalf reverts (54ms)
    with the non ERC20 compliant receiver deployed
      when FEATURE_UNSAFE_TRANSFERS is enabled
        ✓ transfer to unsafe receiver succeeds (56ms)
      when FEATURE_UNSAFE_TRANSFERS is disabled
        ✓ transfer to unsafe receiver reverts (67ms)
      when the receiver is marked as ERC20_RECEIVER
        ✓ transfer to unsafe receiver succeeds (51ms)
      when the receiver is not marked as ERC20_RECEIVER
        ✓ transfer to unsafe receiver reverts (67ms)
      when seder is marked as ERC20_SENDER
        ✓ transfer to unsafe receiver succeeds (59ms)
      when the sender is not marked as ERC20_SENDER
        ✓ transfer to unsafe receiver reverts (82ms)
    when delegation signature is prepared
      when FEATURE_DELEGATIONS_ON_BEHALF is enabled
        ✓ delegation on behalf succeeds (69ms)
      when FEATURE_DELEGATIONS_ON_BEHALF is disabled
        ✓ delegation on behalf reverts (47ms)
    when FEATURE_TRANSFERS is enabled
      ✓ direct transfer succeeds (75ms)
    when FEATURE_TRANSFERS is disabled
      ✓ direct transfer reverts (47ms)
    when FEATURE_OWN_BURNS is enabled
      ✓ self burn succeeds (50ms)
    when FEATURE_OWN_BURNS is disabled
      ✓ self burn reverts (51ms)
    when FEATURE_DELEGATIONS is enabled
      ✓ delegation succeeds (83ms)
    when FEATURE_DELEGATIONS is disabled
      ✓ delegation reverts (44ms)
    when operator is TOKEN_CREATOR
      ✓ mint succeeds (68ms)
    when operator is not TOKEN_CREATOR
      ✓ mint reverts (55ms)
    when operator is TOKEN_DESTROYER
      ✓ burn succeeds (50ms)
    when operator is not TOKEN_DESTROYER
      ✓ burn reverts (56ms)

Contract: DAO (Voting Delegation) tests // Compound
  Run Compound tests
    metadata
      ✓ has given name
      ✓ has given symbol
    balanceOf
      ✓ grants to initial account
    delegateBySig (ILV: delegateWithSig)
      ✓ reverts if the signatory is invalid (55ms)
      ✓ reverts if the nonce is bad  (58ms)
      ✓ reverts if the signature has expired (58ms)
      ✓ delegates on behalf of the signatory (130ms)
    numCheckpoints (ILV: getVotingPowerHistoryLength)
      ✓ returns the number of checkpoints for a delegate (480ms)
      ✓ does not add more than one checkpoint in a block (338ms)
    getPriorVotes (ILV: getVotingPowerAt)
      ✓ reverts if block number >= current block
      ✓ returns 0 if there are no checkpoints
      ✓ returns the latest block if >= last checkpoint block (122ms)
      ✓ returns zero if < first checkpoint block (128ms)
      ✓ generally returns the voting balance at the appropriate checkpoint (515ms)

Contract: IlluviumERC20 Deployment tests
  when deployment arguments are incorrect
    when initial holder H0 is not set (zero)
      ✓ deployment reverts (71ms)
  when deployment arguments are correct
    when H0 is not a deployment account a0
      ✓ H0 doesn't have any permissions
      ✓ token deployment succeeds with the initial token supply S0
      ✓ H0 gets the entire initial balance B0 = S0
      ✓ H0 doesn't have a delegate
      ✓ initial H0 voting power is zero
      ✓ emits Minted event
      ✓ emits Transferred event
      ✓ emits ERC20 Transfer event
    when H0 is a0
      ✓ H0 preservers full permissions bitmask
      ✓ token deployment succeeds with the initial token supply S0
      ✓ H0 gets the entire initial balance B0 = S0
      ✓ H0 doesn't have a delegate
      ✓ initial H0 voting power is zero
      ✓ emits Minted event
      ✓ emits Transferred event
      ✓ emits ERC20 Transfer event

Contract: IlluviumERC20 Functional Requirements tests
  Functional Requirements compliance
    Token Summary
      ✓ Symbol: ILV
      ✓ Name: Illuvium
      ✓ Decimals: 18
      ✓ Initial total supply: 10,000,000 ILV
      ✓ Initial supply holder: H0 - 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b
    Mintable: new tokens may get created
      ✓ not when TOKEN_CREATOR permission is missing (52ms)
      by TOKEN_CREATOR
        tokens get created
          ✓ total supply increases
          ✓ holder balance increases (48ms)
          ✓ Minted event is fired
          ✓ Transferred event is fired
```

```
        ✓ ERC20 Transfer event is fired
    Burnable: existing tokens may get destroyed
      by TOKEN_DESTROYER
        ✓ total supply decreases
        ✓ holder balance decreases
        ✓ Burnt event is fired
        ✓ Transferred event is fired
        ✓ ERC20 Transfer event is fired
      by tokens owner
        ✓ not when OWN_BURNS is disabled (55ms)
        when OWN_BURNS is enabled
          ✓ total supply decreases
          ✓ holder balance decreases
          ✓ Burnt event is fired
          ✓ Transferred event is fired
          ✓ ERC20 Transfer event is fired
      on behalf of tokens owner
        ✓ not when BURNS_ON_BEHALF is disabled (45ms)
        when BURNS_ON_BEHALF is enabled
          ✓ not when token owner didn't approve operation (44ms)
          when token owner approved operation
            ✓ total supply decreases
            ✓ holder balance decreases
            ✓ Burnt event is fired
            ✓ Transferred event is fired
            ✓ ERC20 Transfer event is fired
  Functional Requirements Summary
    Fully ERC20 compliant according to “EIP-20: ERC-20 Token Standard”
      Run Zeppelin ERC20 Tests
        without voting delegation involved
          Zeppelin shouldBehaveLikeERC20
            total supply
              ✓ returns the total amount of tokens
            balanceOf
              when the requested account has no tokens
                ✓ returns zero
              when the requested account has some tokens
                ✓ returns the total amount of tokens
            transfer
              when the recipient is not the zero address
                when the sender does not have enough balance
                  ✓ reverts (46ms)
                when the sender transfers all balance
                  ✓ transfers the requested amount (108ms)
                  ✓ emits a transfer event (44ms)
                when the sender transfers zero tokens
                  ✓ transfers the requested amount (88ms)
                  ✓ emits a transfer event (42ms)
              when the recipient is the zero address
                ✓ reverts (47ms)
            transfer from
              when the token owner is not the zero address
                when the recipient is not the zero address
                  when the spender has enough approved balance
                    when the token owner has enough balance
                      ✓ transfers the requested amount (97ms)
                      ✓ decreases the spender allowance (69ms)
                      ✓ emits a transfer event (51ms)
                      ✓ emits an approval event (72ms)
                    when the token owner does not have enough balance
                      ✓ reverts (52ms)
                  when the spender does not have enough approved balance
                    when the token owner has enough balance
                      ✓ reverts (49ms)
                    when the token owner does not have enough balance
                      ✓ reverts (52ms)
                when the recipient is the zero address
                  ✓ reverts (48ms)
              when the token owner is the zero address
                ✓ reverts (46ms)
            approve
              when the spender is not the zero address
                when the sender has enough balance
                  ✓ emits an approval event (43ms)
                  when there was no approved amount before
                    ✓ approves the requested amount (69ms)
                  when the spender had an approved amount
                    ✓ approves the requested amount and replaces the previous one (64ms)
                when the sender does not have enough balance
                  ✓ emits an approval event (40ms)
                  when there was no approved amount before
                    ✓ approves the requested amount (67ms)
                  when the spender had an approved amount
                    ✓ approves the requested amount and replaces the previous one (66ms)
              when the spender is the zero address
                ✓ reverts (41ms)
            Zeppelin shouldBehaveLikeMint (extracted)
              _mint
                ✓ rejects a null account (43ms)
                for a non zero account
                  ✓ increments totalSupply
                  ✓ increments recipient balance
                  ✓ emits Transfer event
            Zeppelin shouldBehaveLikeBurn (extracted)
              _burn
                ✓ rejects a null account (40ms)
                for a non zero account
                  ✓ rejects burning more than balance (42ms)
                  for entire balance
                    ✓ decrements totalSupply
                    ✓ decrements initialHolder balance
                    ✓ emits Transfer event
                  for less amount than balance
                    ✓ decrements totalSupply
                    ✓ decrements initialHolder balance
                    ✓ emits Transfer event
            Zeppelin shouldBehaveLikeAtomicApprove (extracted)
              decrease allowance
                when the spender is not the zero address
                  when the sender has enough balance
                    when there was no approved amount before
                      ✓ reverts (45ms)
                    when the spender had an approved amount
                      ✓ emits an approval event (68ms)
                      ✓ decreases the spender allowance subtracting the requested amount (68ms)
                      ✓ sets the allowance to zero when all allowance is removed (72ms)
                      ✓ reverts when more than the full allowance is removed (43ms)
                  when the sender does not have enough balance
                    when there was no approved amount before
                      ✓ reverts (43ms)
                    when the spender had an approved amount
                      ✓ emits an approval event (46ms)
                      ✓ decreases the spender allowance subtracting the requested amount (76ms)
                      ✓ sets the allowance to zero when all allowance is removed (77ms)
                      ✓ reverts when more than the full allowance is removed (48ms)
                when the spender is the zero address
                  ✓ reverts (44ms)
              increase allowance
                when the spender is not the zero address
                  when the sender has enough balance
                    ✓ emits an approval event (48ms)
                    when there was no approved amount before
                      ✓ approves the requested amount (72ms)
                    when the spender had an approved amount
                      ✓ increases the spender allowance adding the requested amount (67ms)
                  when the sender does not have enough balance
                    ✓ emits an approval event (50ms)
                    when there was no approved amount before
                      ✓ approves the requested amount (65ms)
                    when the spender had an approved amount
                      ✓ increases the spender allowance adding the requested amount (72ms)
                when the spender is the zero address
                  ✓ reverts (47ms)
          with voting delegation involved
            Zeppelin shouldBehaveLikeERC20
              total supply
                ✓ returns the total amount of tokens
              balanceOf
                when the requested account has no tokens
                  ✓ returns zero
                when the requested account has some tokens
                  ✓ returns the total amount of tokens
              transfer
                when the recipient is not the zero address
                  when the sender does not have enough balance
                    ✓ reverts (50ms)
                  when the sender transfers all balance
                    ✓ transfers the requested amount (109ms)
                    ✓ emits a transfer event (60ms)
                  when the sender transfers zero tokens
                    ✓ transfers the requested amount (106ms)
                    ✓ emits a transfer event (41ms)
                when the recipient is the zero address
                  ✓ reverts (51ms)
              transfer from
                when the token owner is not the zero address
```

```
        when the recipient is not the zero address
          when the spender has enough approved balance
            when the token owner has enough balance
               ✓ transfers the requested amount (101ms)
               ✓ decreases the spender allowance (84ms)
               ✓ emits a transfer event (57ms)
               ✓ emits an approval event (94ms)
            when the token owner does not have enough balance
               ✓ reverts (49ms)
          when the spender does not have enough approved balance
            when the token owner has enough balance
               ✓ reverts (45ms)
            when the token owner does not have enough balance
               ✓ reverts (46ms)
        when the recipient is the zero address
           ✓ reverts (66ms)
      when the token owner is the zero address
         ✓ reverts (51ms)
    approve
      when the spender is not the zero address
        when the sender has enough balance
           ✓ emits an approval event (51ms)
          when there was no approved amount before
             ✓ approves the requested amount (69ms)
          when the spender had an approved amount
             ✓ approves the requested amount and replaces the previous one (70ms)
        when the sender does not have enough balance
           ✓ emits an approval event (41ms)
          when there was no approved amount before
             ✓ approves the requested amount (67ms)
          when the spender had an approved amount
             ✓ approves the requested amount and replaces the previous one (70ms)
      when the spender is the zero address
         ✓ reverts (42ms)
  Zeppelin shouldBehaveLikeMint (extracted)
    _mint
       ✓ rejects a null account (46ms)
      for a non zero account
         ✓ increments totalSupply
         ✓ increments recipient balance
         ✓ emits Transfer event
  Zeppelin shouldBehaveLikeBurn (extracted)
    _burn
       ✓ rejects a null account (41ms)
      for a non zero account
         ✓ rejects burning more than balance (47ms)
        for entire balance
           ✓ decrements totalSupply
           ✓ decrements initialHolder balance
           ✓ emits Transfer event
        for less amount than balance
           ✓ decrements totalSupply
           ✓ decrements initialHolder balance
           ✓ emits Transfer event
  Zeppelin shouldBehaveLikeAtomicApprove (extracted)
    decrease allowance
      when the spender is not the zero address
        when the sender has enough balance
          when there was no approved amount before
             ✓ reverts (42ms)
          when the spender had an approved amount
             ✓ emits an approval event (40ms)
             ✓ decreases the spender allowance subtracting the requested amount (67ms)
             ✓ sets the allowance to zero when all allowance is removed (65ms)
             ✓ reverts when more than the full allowance is removed (45ms)
        when the sender does not have enough balance
          when there was no approved amount before
             ✓ reverts (47ms)
          when the spender had an approved amount
             ✓ emits an approval event (43ms)
             ✓ decreases the spender allowance subtracting the requested amount (67ms)
             ✓ sets the allowance to zero when all allowance is removed (68ms)
             ✓ reverts when more than the full allowance is removed (45ms)
      when the spender is the zero address
         ✓ reverts (45ms)
    increase allowance
      when the spender is not the zero address
        when the sender has enough balance
           ✓ emits an approval event (41ms)
          when there was no approved amount before
             ✓ approves the requested amount (64ms)
          when the spender had an approved amount
             ✓ increases the spender allowance adding the requested amount (95ms)
        when the sender does not have enough balance
           ✓ emits an approval event (52ms)
          when there was no approved amount before
             ✓ approves the requested amount (85ms)
          when the spender had an approved amount
             ✓ increases the spender allowance adding the requested amount (77ms)
      when the spender is the zero address
         ✓ reverts (47ms)
  Initial token supply is minted to an initial holder address
     ✓ initial holder owns total supply (B0 = S0)
  Token holders should be able participate in governance protocol(s) and vote with their tokens
    when delegations are enabled
      when token holder address delegates to itself
         ✓ it becomes a delegate of itself
         ✓ it receives voting power equal to the token balance
         ✓ VotingPowerChanged event is emitted
         ✓ DelegateChanged event is emitted
    otherwise
       ✓ delegation reverts (49ms)
Voting Delegation Requirements
  Token holders posses voting power associated with their tokens
    when there is no tokens on the balance
       ✓ own voting power is zero
    when there are some tokens on the balance
       ✓ own voting power is equal to token balance
  Token holders should be able to act as delegators and to delegate their voting power to any other address - a delegate
    when holder didn't delegate
       ✓ delegate voting power is zero
    when holder delegated
       ✓ delegate voting power is equal to delegator token balance
       ✓ DelegateChanged event is emitted
       ✓ VotingPowerChanged event is emitted
  Any address may become a delegate; delegates are not necessarily token owners
    when delegate has own tokens
       ✓ delegate voting power is sum of delegate and delegator token balances
    when delegate doesn't have own tokens
       ✓ delegate voting power is equal to the delegator token balance
  Voting power delegation doesn't affect token balances
     ✓ delegator voting power is zero
     ✓ delegator token balance remains non-zero
  It should be possible to retrieve voting power of any delegate for any point in time, defined by the Ethereum block number (block height)
     ✓ voting power at any block in 115 blocks range is correct (9543ms)
  Delegators should be able to revoke their voting power delegation
     ✓ delegate's initial voting power is non-zero
    when delegator delegates to the zero address
       ✓ voting power of the previous delegate changes to zero
       ✓ DelegateChanged event is emitted
       ✓ VotingPowerChanged event is emitted
ERC20 Improvements Required
  It should be possible to transfer tokens and invoke a callback function on the receiver in a single transaction (similar to ERC223/ERC777)
    safeTransferFrom behaves safely
      when receiver is an external address
         ✓ sender balance decreases
         ✓ receiver balance increases (42ms)
         ✓ total supply doesn't change
         ✓ emits Transferred event
         ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
           ✓ sender balance decreases
           ✓ receiver balance increases
           ✓ total supply doesn't change
           ✓ emits Transferred event
           ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
           ✓ transfer reverts (61ms)
      when receiver is not an ERC20-compatible smart contract
         ✓ transfer reverts (58ms)
    unsafeTransferFrom behaves unsafely
      when receiver is an external address
         ✓ sender balance decreases
         ✓ receiver balance increases
         ✓ total supply doesn't change
         ✓ emits Transferred event
         ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
           ✓ sender balance decreases
           ✓ receiver balance increases
           ✓ total supply doesn't change
           ✓ emits Transferred event
           ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
```

```
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
    when receiver is not an ERC20-compatible smart contract
      ✓ sender balance decreases
      ✓ receiver balance increases
      ✓ total supply doesn't change
      ✓ emits Transferred event
      ✓ emits ERC20 Transfer event
  transferFrom
    when UNSAFE_TRANSFERS are enabled
      when receiver is an external address
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      when receiver is not an ERC20-compatible smart contract
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
    when token receiver is ERC20_RECEIVER
      when receiver is an external address
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      when receiver is not an ERC20-compatible smart contract
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
    when token sender is ERC20_SENDER
      when receiver is an external address
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      when receiver is not an ERC20-compatible smart contract
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
    otherwise
      when receiver is an external address
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
          ✓ transfer reverts (65ms)
      when receiver is not an ERC20-compatible smart contract
        ✓ transfer reverts (59ms)
  Support for atomic allowance modification, resolution of well-known ERC20 issue with approve
    decrease allowance
      when the spender is not the zero address
        when the sender has enough balance
          when there was no approved amount before
            ✓ reverts (47ms)
          when the spender had an approved amount
            ✓ emits an approval event (62ms)
            ✓ decreases the spender allowance subtracting the requested amount (70ms)
            ✓ sets the allowance to zero when all allowance is removed (70ms)
            ✓ reverts when more than the full allowance is removed (43ms)
        when the sender does not have enough balance
          when there was no approved amount before
            ✓ reverts (46ms)
          when the spender had an approved amount
            ✓ emits an approval event (42ms)
            ✓ decreases the spender allowance subtracting the requested amount (66ms)
            ✓ sets the allowance to zero when all allowance is removed (76ms)
            ✓ reverts when more than the full allowance is removed (50ms)
      when the spender is the zero address
        ✓ reverts (46ms)
    increase allowance
      when the spender is not the zero address
        when the sender has enough balance
          ✓ emits an approval event (46ms)
          when there was no approved amount before
            ✓ approves the requested amount (73ms)
          when the spender had an approved amount
            ✓ increases the spender allowance adding the requested amount (80ms)
        when the sender does not have enough balance
          ✓ emits an approval event (42ms)
          when there was no approved amount before
            ✓ approves the requested amount (71ms)
          when the spender had an approved amount
            ✓ increases the spender allowance adding the requested amount (67ms)
      when the spender is the zero address
        ✓ reverts (48ms)

Contract: IlluviumERC20 Mint/Burn tests
  Minting/Burning
    Minting
      Minting
        ✓ when performed not by TOKEN_CREATOR - mint reverts (48ms)
        when performed by TOKEN_CREATOR
          ✓ when the recipient is zero address - mint reverts (51ms)
          ✓ when amount is too big and causes total supply overflow - mint reverts (42ms)
          otherwise (when recipient and amount are valid)
            ✓ total supply increases
            ✓ recipient balance increases
            ✓ emits Minted event
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
    Burning
      when performed by TOKEN_DESTROYER
        ✓ when the amount is zero - burn reverts (40ms)
        ✓ when supplier address is zero address - burn reverts (48ms)
        ✓ when supplier doesn't have enough tokens - burn reverts (41ms)
        when amount and supplier address are correct
```

```
            ✓ total supply decreases
            ✓ supplier balance decreases
            ✓ emits Burnt event
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
        when performed not by TOKEN_DESTROYER
          when burning own tokens
            when OWN_BURNS is enabled
               ✓ when the amount is zero - burn reverts (41ms)
               ✓ when supplier address is zero address - burn reverts (46ms)
               ✓ when supplier doesn't have enough tokens - burn reverts (48ms)
              when amount and supplier address are correct
                 ✓ total supply decreases
                 ✓ supplier balance decreases
                 ✓ emits Burnt event
                 ✓ emits Transferred event
                 ✓ emits ERC20 Transfer event
            when OWN_BURNS is not enabled
               ✓ burn reverts (46ms)
          when burning tokens on behalf
            when BURNS_ON_BEHALF is enabled
               ✓ when the amount is zero - burn reverts (47ms)
               ✓ when supplier address is zero address - burn reverts (43ms)
               ✓ when supplier doesn't have enough tokens - burn reverts (41ms)
              when amount and supplier address are correct
                 ✓ total supply decreases
                 ✓ supplier balance decreases
                 ✓ emits Burnt event
                 ✓ emits Transferred event
                 ✓ emits ERC20 Transfer event
            when BURNS_ON_BEHALF is not enabled
               ✓ burn reverts (43ms)
          otherwise (unauthorized burn)
             ✓ burn reverts (42ms)

  Contract: IlluviumERC20 Non-functional Requirements tests
    Non-functional Requirements compliance
      Gas Consumption
        deployment
           ✓ consumes no more than 2523840 gas
        approve
           ✓ consumes no more than 47183 gas
        atomic approve (increase)
           ✓ consumes no more than 48322 gas
        atomic approve (decrease)
           ✓ consumes no more than 33344 gas
        when delegation is not involved
          direct transfer
             ✓ consumes no more than 61368 gas
          transfer on behalf
             ✓ consumes no more than 72154 gas
          mint
             ✓ consumes no more than 58920 gas
          burn
             ✓ consumes no more than 44269 gas
          burn on behalf
             ✓ consumes no more than 54613 gas
        when first address is a delegate
          direct transfer
             ✓ consumes no more than 97278 gas
          transfer on behalf
             ✓ consumes no more than 108061 gas
        when second address is a delegate
          direct transfer
             ✓ consumes no more than 107099 gas
          transfer on behalf
             ✓ consumes no more than 117882 gas
        when delegation is fully involved
          direct transfer
             ✓ consumes no more than 142933 gas
          transfer on behalf
             ✓ consumes no more than 153716 gas
          mint
             ✓ consumes no more than 104651 gas
          burn
             ✓ consumes no more than 80179 gas
          burn on behalf
             ✓ consumes no more than 90520 gas
        when there is nothing on the balances
          delegate
             ✓ consumes no more than 46803 gas
          delegate on behalf (with sig)
             ✓ consumes no more than 75116 gas
        when one of the balances is non-zero
          delegate
             ✓ consumes no more than 92532 gas
          delegate on behalf (with sig)
             ✓ consumes no more than 120830 gas
        when both balances are non-zero
          delegate
             ✓ consumes no more than 113366 gas
          delegate on behalf (with sig)
             ✓ consumes no more than 126676 gas

  Contract: IlluviumERC20 Quantstamp Audit tests
    Quantstamp Audit
      QSP-2 Wrong underflow check
        when allowance is set to MAX_UINT256
           ✓ allowance is MAX_UINT256
           ✓ decreaseAllowance by zero fails (50ms)
          decreaseAllowance by 1 succeeds without overflow/underflow
             ✓ allowance is MAX_UINT256 - 1
          decreaseAllowance by MAX_UINT256 succeeds without overflow/underflow
             ✓ allowance is zero

  Contract: IlluviumERC20 Unit Tests
    when token is not deployed (deployment routine)
      when deployment arguments are incorrect
        when initial holder H0 is not set (zero)
           ✓ deployment reverts (115ms)
      when deployment arguments are correct
        when H0 is not a deployment account a0
           ✓ H0 doesn't have any permissions
           ✓ token deployment succeeds with the initial token supply S0
           ✓ H0 gets the entire initial balance B0 = S0
           ✓ H0 doesn't have a delegate
           ✓ initial H0 voting power is zero
           ✓ emits Minted event
           ✓ emits Transferred event
           ✓ emits ERC20 Transfer event
        when H0 is a0
           ✓ H0 preservers full permissions bitmask
           ✓ token deployment succeeds with the initial token supply S0
           ✓ H0 gets the entire initial balance B0 = S0
           ✓ H0 doesn't have a delegate
           ✓ initial H0 voting power is zero
           ✓ emits Minted event
           ✓ emits Transferred event
           ✓ emits ERC20 Transfer event
    when token is deployed (usage routines)
      ACL Core
        when performed by ACCESS_MANAGER
          when ACCESS_MANAGER has full set of permissions
            what you set
               ✓ is what you get
            what you remove
               ✓ is what gets removed
          when ACCESS_MANAGER doesn't have any permissions
            what you get, independently of what you set
               ✓ is always zero
            what you get, independently of what you remove
               ✓ is always what you had
          when ACCESS_MANAGER has some permissions
            what you get
               ✓ is an intersection of what you set and what you have
            what you remove
               ✓ is an intersection of what you tried to remove and what you have
          ACCESS_MANAGER updates itself
             ✓ and degrades to zero (502ms)
          when ACCESS_MANAGER grants ACCESS_MANAGER permission
             ✓ operator becomes an ACCESS_MANAGER (46ms)
          when ACCESS_MANAGER revokes ACCESS_MANAGER permission from itself
             ✓ operator becomes an ACCESS_MANAGER (44ms)
        otherwise (no ACCESS_MANAGER permission)
           ✓ updateFeatures reverts (40ms)
           ✓ updateRole reverts (40ms)
      ACL Illuvium ERC20
        after spender's approval
          when FEATURE_TRANSFERS_ON_BEHALF is enabled
             ✓ transfer on behalf succeeds (45ms)
          when FEATURE_TRANSFERS_ON_BEHALF is disabled
             ✓ transfer on behalf reverts (40ms)
          when FEATURE_BURNS_ON_BEHALF is enabled
             ✓ burn on behalf succeeds (42ms)
          when FEATURE_BURNS_ON_BEHALF is disabled
             ✓ burn on behalf reverts (47ms)
```

```
    with the non ERC20 compliant receiver deployed
      when FEATURE_UNSAFE_TRANSFERS is enabled
        ✓ transfer to unsafe receiver succeeds (42ms)
      when FEATURE_UNSAFE_TRANSFERS is disabled
        ✓ transfer to unsafe receiver reverts (56ms)
      when the receiver is marked as ERC20_RECEIVER
        ✓ transfer to unsafe receiver succeeds (48ms)
      when the receiver is not marked as ERC20_RECEIVER
        ✓ transfer to unsafe receiver reverts (57ms)
      when seder is marked as ERC20_SENDER
        ✓ transfer to unsafe receiver succeeds (52ms)
      when the sender is not marked as ERC20_SENDER
        ✓ transfer to unsafe receiver reverts (61ms)
    when delegation signature is prepared
      when FEATURE_DELEGATIONS_ON_BEHALF is enabled
        ✓ delegation on behalf succeeds (48ms)
      when FEATURE_DELEGATIONS_ON_BEHALF is disabled
        ✓ delegation on behalf reverts (41ms)
    when FEATURE_TRANSFERS is enabled
      ✓ direct transfer succeeds (44ms)
    when FEATURE_TRANSFERS is disabled
      ✓ direct transfer reverts (42ms)
    when FEATURE_OWN_BURNS is enabled
      ✓ self burn succeeds (47ms)
    when FEATURE_OWN_BURNS is disabled
      ✓ self burn reverts (47ms)
    when FEATURE_DELEGATIONS is enabled
      ✓ delegation succeeds (43ms)
    when FEATURE_DELEGATIONS is disabled
      ✓ delegation reverts (42ms)
    when operator is TOKEN_CREATOR
      ✓ mint succeeds (49ms)
    when operator is not TOKEN_CREATOR
      ✓ mint reverts (44ms)
    when operator is TOKEN_DESTROYER
      ✓ burn succeeds (42ms)
    when operator is not TOKEN_DESTROYER
      ✓ burn reverts (45ms)
  Functional Requirements compliance
    Token Summary
      ✓ Symbol: ILV
      ✓ Name: Illuvium
      ✓ Decimals: 18
      ✓ Initial total supply: 10,000,000 ILV
      ✓ Initial supply holder: H0 - 0x22d491Bde2303f2f43325b2108D26f1eAbA1e32b
    Mintable: new tokens may get created
      ✓ not when TOKEN_CREATOR permission is missing (43ms)
      by TOKEN_CREATOR
        tokens get created
          ✓ total supply increases
          ✓ holder balance increases
          ✓ Minted event is fired
          ✓ Transferred event is fired
          ✓ ERC20 Transfer event is fired
    Burnable: existing tokens may get destroyed
      by TOKEN_DESTROYER
        ✓ total supply decreases
        ✓ holder balance decreases
        ✓ Burnt event is fired
        ✓ Transferred event is fired
        ✓ ERC20 Transfer event is fired
      by tokens owner
        ✓ not when OWN_BURNS is disabled (45ms)
        when OWN_BURNS is enabled
          ✓ total supply decreases
          ✓ holder balance decreases
          ✓ Burnt event is fired
          ✓ Transferred event is fired
          ✓ ERC20 Transfer event is fired
      on behalf of tokens owner
        ✓ not when BURNS_ON_BEHALF is disabled (47ms)
        when BURNS_ON_BEHALF is enabled
          ✓ not when token owner didn't approve operation (45ms)
          when token owner approved operation
            ✓ total supply decreases
            ✓ holder balance decreases
            ✓ Burnt event is fired
            ✓ Transferred event is fired
            ✓ ERC20 Transfer event is fired
    Functional Requirements Summary
      Fully ERC20 compliant according to “EIP-20: ERC-20 Token Standard”
      Run Zeppelin ERC20 Tests
        without voting delegation involved
          Zeppelin shouldBehaveLikeERC20
            total supply
              ✓ returns the total amount of tokens
            balanceOf
              when the requested account has no tokens
                ✓ returns zero
              when the requested account has some tokens
                ✓ returns the total amount of tokens
            transfer
              when the recipient is not the zero address
                when the sender does not have enough balance
                  ✓ reverts (63ms)
                when the sender transfers all balance
                  ✓ transfers the requested amount (92ms)
                  ✓ emits a transfer event (44ms)
                when the sender transfers zero tokens
                  ✓ transfers the requested amount (88ms)
                  ✓ emits a transfer event (46ms)
              when the recipient is the zero address
                ✓ reverts (44ms)
            transfer from
              when the token owner is not the zero address
                when the recipient is not the zero address
                  when the spender has enough approved balance
                    when the token owner has enough balance
                      ✓ transfers the requested amount (98ms)
                      ✓ decreases the spender allowance (81ms)
                      ✓ emits a transfer event (46ms)
                      ✓ emits an approval event (69ms)
                    when the token owner does not have enough balance
                      ✓ reverts (55ms)
                  when the spender does not have enough approved balance
                    when the token owner has enough balance
                      ✓ reverts (49ms)
                    when the token owner does not have enough balance
                      ✓ reverts (46ms)
                when the recipient is the zero address
                  ✓ reverts (45ms)
              when the token owner is the zero address
                ✓ reverts (45ms)
            approve
              when the spender is not the zero address
                when the sender has enough balance
                  ✓ emits an approval event (43ms)
                when there was no approved amount before
                  ✓ approves the requested amount (85ms)
                when the spender had an approved amount
                  ✓ approves the requested amount and replaces the previous one (68ms)
              when the sender does not have enough balance
                ✓ emits an approval event (41ms)
                when there was no approved amount before
                  ✓ approves the requested amount (66ms)
                when the spender had an approved amount
                  ✓ approves the requested amount and replaces the previous one (77ms)
            when the spender is the zero address
              ✓ reverts (54ms)
          Zeppelin shouldBehaveLikeMint (extracted)
            _mint
              ✓ rejects a null account (47ms)
              for a non zero account
                ✓ increments totalSupply
                ✓ increments recipient balance
                ✓ emits Transfer event
          Zeppelin shouldBehaveLikeBurn (extracted)
            _burn
              ✓ rejects a null account (45ms)
              for a non zero account
                ✓ rejects burning more than balance (45ms)
                for entire balance
                  ✓ decrements totalSupply
                  ✓ decrements initialHolder balance
                  ✓ emits Transfer event
                for less amount than balance
                  ✓ decrements totalSupply
                  ✓ decrements initialHolder balance
                  ✓ emits Transfer event
          Zeppelin shouldBehaveLikeAtomicApprove (extracted)
            decrease allowance
              when the spender is not the zero address
                when the sender has enough balance
                  when there was no approved amount before
                    ✓ reverts (49ms)
                  when the spender had an approved amount
```

```
                ✓ emits an approval event (46ms)
                ✓ decreases the spender allowance subtracting the requested amount (73ms)
                ✓ sets the allowance to zero when all allowance is removed (76ms)
                ✓ reverts when more than the full allowance is removed (43ms)
            when the sender does not have enough balance
              when there was no approved amount before
                ✓ reverts (43ms)
              when the spender had an approved amount
                ✓ emits an approval event (50ms)
                ✓ decreases the spender allowance subtracting the requested amount (68ms)
                ✓ sets the allowance to zero when all allowance is removed (76ms)
                ✓ reverts when more than the full allowance is removed (54ms)
          when the spender is the zero address
            ✓ reverts (45ms)
        increase allowance
          when the spender is not the zero address
            when the sender has enough balance
              ✓ emits an approval event (71ms)
              when there was no approved amount before
                ✓ approves the requested amount (85ms)
              when the spender had an approved amount
                ✓ increases the spender allowance adding the requested amount (87ms)
            when the sender does not have enough balance
              ✓ emits an approval event (55ms)
              when there was no approved amount before
                ✓ approves the requested amount (79ms)
              when the spender had an approved amount
                ✓ increases the spender allowance adding the requested amount (68ms)
          when the spender is the zero address
            ✓ reverts (45ms)
    with voting delegation involved
      Zeppelin shouldBehaveLikeERC20
        total supply
          ✓ returns the total amount of tokens
        balanceOf
          when the requested account has no tokens
            ✓ returns zero
          when the requested account has some tokens
            ✓ returns the total amount of tokens
        transfer
          when the recipient is not the zero address
            when the sender does not have enough balance
              ✓ reverts (47ms)
            when the sender transfers all balance
              ✓ transfers the requested amount (109ms)
              ✓ emits a transfer event (63ms)
            when the sender transfers zero tokens
              ✓ transfers the requested amount (106ms)
              ✓ emits a transfer event (42ms)
          when the recipient is the zero address
            ✓ reverts (45ms)
        transfer from
          when the token owner is not the zero address
            when the recipient is not the zero address
              when the spender has enough approved balance
                when the token owner has enough balance
                  ✓ transfers the requested amount (106ms)
                  ✓ decreases the spender allowance (80ms)
                  ✓ emits a transfer event (69ms)
                  ✓ emits an approval event (93ms)
                when the token owner does not have enough balance
                  ✓ reverts (66ms)
              when the spender does not have enough approved balance
                when the token owner has enough balance
                  ✓ reverts (88ms)
                when the token owner does not have enough balance
                  ✓ reverts (66ms)
            when the recipient is the zero address
              ✓ reverts (102ms)
          when the token owner is the zero address
            ✓ reverts (72ms)
        approve
          when the spender is not the zero address
            when the sender has enough balance
              ✓ emits an approval event (46ms)
              when there was no approved amount before
                ✓ approves the requested amount (132ms)
              when the spender had an approved amount
                ✓ approves the requested amount and replaces the previous one (91ms)
            when the sender does not have enough balance
              ✓ emits an approval event (56ms)
              when there was no approved amount before
                ✓ approves the requested amount (82ms)
              when the spender had an approved amount
                ✓ approves the requested amount (82ms)
              when the spender had an approved amount
                ✓ approves the requested amount and replaces the previous one (86ms)
            when the spender is the zero address
              ✓ reverts (43ms)
      Zeppelin shouldBehaveLikeMint (extracted)
        _mint
          ✓ rejects a null account (63ms)
          for a non zero account
            ✓ increments totalSupply
            ✓ increments recipient balance
            ✓ emits Transfer event
      Zeppelin shouldBehaveLikeBurn (extracted)
        _burn
          ✓ rejects a null account (75ms)
          for a non zero account
            ✓ rejects burning more than balance (47ms)
            for entire balance
              ✓ decrements totalSupply
              ✓ decrements initialHolder balance
              ✓ emits Transfer event
            for less amount than balance
              ✓ decrements totalSupply
              ✓ decrements initialHolder balance
              ✓ emits Transfer event
      Zeppelin shouldBehaveLikeAtomicApprove (extracted)
        decrease allowance
          when the spender is not the zero address
            when the sender has enough balance
              when there was no approved amount before
                ✓ reverts (50ms)
              when the spender had an approved amount
                ✓ emits an approval event (46ms)
                ✓ decreases the spender allowance subtracting the requested amount (84ms)
                ✓ sets the allowance to zero when all allowance is removed (65ms)
                ✓ reverts when more than the full allowance is removed (44ms)
            when the sender does not have enough balance
              when there was no approved amount before
                ✓ reverts (47ms)
              when the spender had an approved amount
                ✓ emits an approval event (50ms)
                ✓ decreases the spender allowance subtracting the requested amount (68ms)
                ✓ sets the allowance to zero when all allowance is removed (86ms)
                ✓ reverts when more than the full allowance is removed (39ms)
          when the spender is the zero address
            ✓ reverts (44ms)
        increase allowance
          when the spender is not the zero address
            when the sender has enough balance
              ✓ emits an approval event (43ms)
              when there was no approved amount before
                ✓ approves the requested amount (68ms)
              when the spender had an approved amount
                ✓ increases the spender allowance adding the requested amount (66ms)
            when the sender does not have enough balance
              ✓ emits an approval event (43ms)
              when there was no approved amount before
                ✓ approves the requested amount (67ms)
              when the spender had an approved amount
                ✓ increases the spender allowance adding the requested amount (66ms)
          when the spender is the zero address
            ✓ reverts (40ms)
    Initial token supply is minted to an initial holder address
      ✓ initial holder owns total supply (B0 = S0)
    Token holders should be able participate in governance protocol(s) and vote with their tokens
      when delegations are enabled
        when token holder address delegates to itself
          ✓ it becomes a delegate of itself
          ✓ it receives voting power equal to the token balance
          ✓ VotingPowerChanged event is emitted
          ✓ DelegateChanged event is emitted
      otherwise
        ✓ delegation reverts (58ms)
  Voting Delegation Requirements
    Token holders posses voting power associated with their tokens
      when there is no tokens on the balance
        ✓ own voting power is zero
      when there are some tokens on the balance
        ✓ own voting power is equal to token balance
    Token holders should be able to act as delegators and to delegate their voting power to any other address - a delegate
      when holder didn't delegate
        ✓ delegate voting power is zero
      when holder delegated
        ✓ delegate voting power is equal to delegator token balance
```

```
      ✓ DelegateChanged event is emitted
      ✓ VotingPowerChanged event is emitted
  Any address may become a delegate; delegates are not necessarily token owners
    when delegate has own tokens
      ✓ delegate voting power is sum of delegate and delegator token balances
    when delegate doesn't have own tokens
      ✓ delegate voting power is equal to the delegator token balance
  Voting power delegation doesn’t affect token balances
    ✓ delegator voting power is zero
    ✓ delegator token balance remains non-zero
  It should be possible to retrieve voting power of any delegate for any point in time, defined by the Ethereum block number (block height)
    ✓ voting power at any block in 115 blocks range is correct (9841ms)
  Delegators should be able to revoke their voting power delegation
    ✓ delegate's initial voting power is non-zero
    when delegator delegates to the zero address
      ✓ voting power of the previous delegate changes to zero
      ✓ DelegateChanged event is emitted
      ✓ VotingPowerChanged event is emitted
ERC20 Improvements Required
  It should be possible to transfer tokens and invoke a callback function on the receiver in a single transaction (similar to ERC223/ERC777)
    safeTransferFrom behaves safely
      when receiver is an external address
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
          ✓ transfer reverts (60ms)
      when receiver is not an ERC20-compatible smart contract
        ✓ transfer reverts (57ms)
    unsafeTransferFrom behaves unsafely
      when receiver is an external address
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
      when receiver is an ERC20-compatible smart contract (ERC20Receiver)
        when receiver responds onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver doesn't respond onERC20Received
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      when receiver is not an ERC20-compatible smart contract
        ✓ sender balance decreases
        ✓ receiver balance increases
        ✓ total supply doesn't change
        ✓ emits Transferred event
        ✓ emits ERC20 Transfer event
    transferFrom
      when UNSAFE_TRANSFERS are enabled
        when receiver is an external address
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver is an ERC20-compatible smart contract (ERC20Receiver)
          when receiver responds onERC20Received
            ✓ sender balance decreases
            ✓ receiver balance increases
            ✓ total supply doesn't change
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
          when receiver doesn't respond onERC20Received
            ✓ sender balance decreases
            ✓ receiver balance increases
            ✓ total supply doesn't change
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
        when receiver is not an ERC20-compatible smart contract
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      when token receiver is ERC20_RECEIVER
        when receiver is an external address
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver is an ERC20-compatible smart contract (ERC20Receiver)
          when receiver responds onERC20Received
            ✓ sender balance decreases
            ✓ receiver balance increases
            ✓ total supply doesn't change
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
          when receiver doesn't respond onERC20Received
            ✓ sender balance decreases
            ✓ receiver balance increases
            ✓ total supply doesn't change
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
        when receiver is not an ERC20-compatible smart contract
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      when token sender is ERC20_SENDER
        when receiver is an external address
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver is an ERC20-compatible smart contract (ERC20Receiver)
          when receiver responds onERC20Received
            ✓ sender balance decreases
            ✓ receiver balance increases
            ✓ total supply doesn't change
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
          when receiver doesn't respond onERC20Received
            ✓ sender balance decreases
            ✓ receiver balance increases
            ✓ total supply doesn't change
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
        when receiver is not an ERC20-compatible smart contract
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      otherwise
        when receiver is an external address
          ✓ sender balance decreases
          ✓ receiver balance increases
          ✓ total supply doesn't change
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
        when receiver is an ERC20-compatible smart contract (ERC20Receiver)
          when receiver responds onERC20Received
            ✓ sender balance decreases
            ✓ receiver balance increases
            ✓ total supply doesn't change
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
          when receiver doesn't respond onERC20Received
            ✓ transfer reverts (96ms)
        when receiver is not an ERC20-compatible smart contract
          ✓ transfer reverts (64ms)
  Support for atomic allowance modification, resolution of well-known ERC20 issue with approve
    decrease allowance
      when the spender is not the zero address
        when the sender has enough balance
          when there was no approved amount before
```

```
                ✓ reverts (47ms)
              when the spender had an approved amount
                ✓ emits an approval event (44ms)
                ✓ decreases the spender allowance subtracting the requested amount (73ms)
                ✓ sets the allowance to zero when all allowance is removed (75ms)
                ✓ reverts when more than the full allowance is removed (44ms)
            when the sender does not have enough balance
              when there was no approved amount before
                ✓ reverts (42ms)
              when the spender had an approved amount
                ✓ emits an approval event (42ms)
                ✓ decreases the spender allowance subtracting the requested amount (67ms)
                ✓ sets the allowance to zero when all allowance is removed (68ms)
                ✓ reverts when more than the full allowance is removed (44ms)
          when the spender is the zero address
            ✓ reverts (43ms)
        increase allowance
          when the spender is not the zero address
            when the sender has enough balance
              ✓ emits an approval event (44ms)
            when there was no approved amount before
              ✓ approves the requested amount (73ms)
            when the spender had an approved amount
              ✓ increases the spender allowance adding the requested amount (66ms)
            when the sender does not have enough balance
              ✓ emits an approval event (43ms)
            when there was no approved amount before
              ✓ approves the requested amount (74ms)
            when the spender had an approved amount
              ✓ increases the spender allowance adding the requested amount (63ms)
          when the spender is the zero address
            ✓ reverts (40ms)
  Non-functional Requirements compliance
    Gas Consumption
      deployment
        ✓ consumes no more than 2523840 gas
      approve
        ✓ consumes no more than 47183 gas
      atomic approve (increase)
        ✓ consumes no more than 48322 gas
      atomic approve (decrease)
        ✓ consumes no more than 33344 gas
      when delegation is not involved
        direct transfer
          ✓ consumes no more than 61368 gas
        transfer on behalf
          ✓ consumes no more than 72154 gas
        mint
          ✓ consumes no more than 58920 gas
        burn
          ✓ consumes no more than 44269 gas
        burn on behalf
          ✓ consumes no more than 54613 gas
      when first address is a delegate
        direct transfer
          ✓ consumes no more than 97278 gas
        transfer on behalf
          ✓ consumes no more than 108061 gas
      when second address is a delegate
        direct transfer
          ✓ consumes no more than 107099 gas
        transfer on behalf
          ✓ consumes no more than 117882 gas
      when delegation is fully involved
        direct transfer
          ✓ consumes no more than 142933 gas
        transfer on behalf
          ✓ consumes no more than 153716 gas
        mint
          ✓ consumes no more than 104651 gas
        burn
          ✓ consumes no more than 80179 gas
        burn on behalf
          ✓ consumes no more than 90520 gas
      when there is nothing on the balances
        delegate
          ✓ consumes no more than 46803 gas
        delegate on behalf (with sig)
          ✓ consumes no more than 75116 gas
      when one of the balances is non-zero
        delegate
          ✓ consumes no more than 92532 gas
        delegate on behalf (with sig)
          ✓ consumes no more than 120830 gas
      when both balances are non-zero
        delegate
          ✓ consumes no more than 113366 gas
        delegate on behalf (with sig)
          ✓ consumes no more than 126676 gas
  Minting/Burning
    Minting
      Minting
        ✓ when performed not by TOKEN_CREATOR - mint reverts (46ms)
        when performed by TOKEN_CREATOR
          ✓ when the recipient is zero address - mint reverts (46ms)
          ✓ when amount is too big and causes total supply overflow - mint reverts (45ms)
          otherwise (when recipient and amount are valid)
            ✓ total supply increases
            ✓ recipient balance increases
            ✓ emits Minted event
            ✓ emits Transferred event
            ✓ emits ERC20 Transfer event
    Burning
      when performed by TOKEN_DESTROYER
        ✓ when the amount is zero - burn reverts (43ms)
        ✓ when supplier address is zero address - burn reverts (46ms)
        ✓ when supplier doesn't have enough tokens - burn reverts (45ms)
        when amount and supplier address are correct
          ✓ total supply decreases
          ✓ supplier balance decreases
          ✓ emits Burnt event
          ✓ emits Transferred event
          ✓ emits ERC20 Transfer event
      when performed not by TOKEN_DESTROYER
        when burning own tokens
          when OWN_BURNS is enabled
            ✓ when the amount is zero - burn reverts (54ms)
            ✓ when supplier address is zero address - burn reverts (53ms)
            ✓ when supplier doesn't have enough tokens - burn reverts (46ms)
            when amount and supplier address are correct
              ✓ total supply decreases
              ✓ supplier balance decreases
              ✓ emits Burnt event
              ✓ emits Transferred event
              ✓ emits ERC20 Transfer event
          when OWN_BURNS is not enabled
            ✓ burn reverts (51ms)
        when burning tokens on behalf
          when BURNS_ON_BEHALF is enabled
            ✓ when the amount is zero - burn reverts (46ms)
            ✓ when supplier address is zero address - burn reverts (44ms)
            ✓ when supplier doesn't have enough tokens - burn reverts (46ms)
            when amount and supplier address are correct
              ✓ total supply decreases
              ✓ supplier balance decreases (57ms)
              ✓ emits Burnt event
              ✓ emits Transferred event
              ✓ emits ERC20 Transfer event
          when BURNS_ON_BEHALF is not enabled
            ✓ burn reverts (59ms)
        otherwise (unauthorized burn)
          ✓ burn reverts (85ms)
    Quantstamp Audit
      QSP-2 Wrong underflow check
        when allowance is set to MAX_UINT256
          ✓ allowance is MAX_UINT256 (64ms)
          ✓ decreaseAllowance by zero fails (100ms)
        decreaseAllowance by 1 succeeds without overflow/underflow
          ✓ allowance is MAX_UINT256 - 1
        decreaseAllowance by MAX_UINT256 succeeds without overflow/underflow
          ✓ allowance is zero
  Run Compound tests
    metadata
      ✓ has given name
      ✓ has given symbol
    balanceOf
      ✓ grants to initial account
    delegateBySig (ILV: delegateWithSig)
      ✓ reverts if the signatory is invalid (60ms)
      ✓ reverts if the nonce is bad  (63ms)
      ✓ reverts if the signature has expired (76ms)
      ✓ delegates on behalf of the signatory (112ms)
    numCheckpoints (ILV: getVotingPowerHistoryLength)
      ✓ returns the number of checkpoints for a delegate (531ms)
      ✓ does not add more than one checkpoint in a block (395ms)
    getPriorVotes (ILV: getVotingPowerAt)
```

```
        ✓ reverts if block number >= current block
        ✓ returns 0 if there are no checkpoints
        ✓ returns the latest block if >= last checkpoint block (138ms)
        ✓ returns zero if < first checkpoint block (146ms)
        ✓ generally returns the voting balance at the appropriate checkpoint (742ms)


  818 passing (4m)
```

# 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

8f1fdee146be83108fb7cf8d0f54c88d72998da3198fe725bece06bef65703a2 ./AddressUtils.sol

9e06ea99bbad80a7c86c3b5ed2d3c67b1fde4445f693146072a78e3a121e9c41 ./AccessControl.sol

b7c53d8d99183bbd39e10607a0264760073c652a4f50ffe57cad11a596fa5267 ./token/IlluviumERC20.sol

d653a70303ca168c629a9bb437988f8021ba47170ca1a2c924706dd644ac9cbc ./token/ERC20Receiver.sol

### Tests

f7793b30bb208823885cf747639ace13b190bc2149951a7506358c2fe8621a00 ./test/ilv_erc20_unit.js

cf613d285d7fb6fcc0ae397932c03e74fb2c940786e48f9fc2adc83852d27a62 ./test/ilv_erc20_deployment.js

ddd4e7cb72579821b9cef4c75866b15ed8980a9981e4cc489c83bc472db63e89 ./test/ilv_erc20_non_func.js

d855137efafb613e9fdc5414bc2e564725ba7ad3d4c2a9177a72d4a5832d8ebd ./test/ilv_erc20_func_req.js

32df8e24895781bb13b94e96321492f8119e3ac73dd7281418b1e3460a8a634d ./test/ilv_erc20_qs_audit.js

5ba491dac4b172d73117c762aa4bf71c620f4688e084cb3c7670ece2b2878550 ./test/ilv_erc20_acl.js

776558fa65ba3fa7b74f979daee42af16d3c252556e1bfbbbc96d44554e65605 ./test/ilv_erc20_mint_burn.js

f59e3848bf93607f684a19bade61b16691d1c2a43a440474e1d0ac126f1d4843 ./test/ilv_erc20_dao.js

# Changelog

- 2021-03-16 - Initial report
- 2021-03-19 - Final report based on commit 903a787

## About Quantstamp

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

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment 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.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.


#### Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

#### Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

#### Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp, Inc. (Quantstamp). Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

#### Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.