HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Tokeny
Date: March 30th, 2023

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for Tokeny
Approved By	Evgeniy Bezuglyi | SC Audits Department Head at Hacken OU
Type	ERC20 token
Platform	EVM
Language	Solidity
Methodology	Link
Website	https://tokeny.com/
Changelog	06.12.2022 - Initial Review 30.12.2022 - Second Review 30.03.2023 - Third Review

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by Tokeny (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:

Initial review scope

Repository	https://github.com/TokenySolutions/T-REX/tree/trex_4
Commit	5eb742a968219cdde6fb58a95ebbe68c9b4ea5af
Whitepaper	https://tokeny.com/wp-content/uploads/2020/05/Whitepaper-T-REX-Security-Tokens-V3.pdf
Functional Requirements	https://docs.tokeny.com/docs/smart-contracts
Technical Requirements	https://docs.tokeny.com/docs/smart-contracts
Contracts Addresses	Not deployed
Contracts	File: ./contracts/compliance/modular/IModularCompliance.sol SHA3: 90641d5030eda088c310b7e3f699bb2030a9ecf601882106c4b06cc3a6214b8e File: ./contracts/compliance/modular/MCStorage.sol SHA3: afe73986cd488efd65761fc5f5300bd284d7a862c8972df3be38bd82384c0161 File: ./contracts/compliance/modular/ModularCompliance.sol SHA3: c9f89c22b8b3e915401ca1c8e0e813d683fcb5124d2b7f02575f6e33ed0b6996 File: ./contracts/compliance/modular/modules/AbstractModule.sol SHA3: c522dad54aff00ca0864537bda15531a4c7ac185b21ee80b07357b50a2f3b2d9 File: ./contracts/compliance/modular/modules/CountryRestrictModule.sol SHA3: f69993c4bd604aaf833740f185bacd065a8f22d393cb526ad9c6460d07a8129a File: ./contracts/compliance/modular/modules/IModule.sol SHA3: 21a29400077d164b310f7683bcb0121e03c168792cbbf7fa82cd0ccc021c6872 File: ./contracts/factory/ITREXFactory.sol SHA3: a76ba374f097baa5b4c3dffb1de95f23a20a777803abe4bbc4e621d50281c66d File: ./contracts/factory/TREXFactory.sol SHA3: 729b490bbabed3b96a19a37af05c6e122a9e4bb138f5c6b598ae5bc298bf9987 File: ./contracts/proxy/authority/ITREXImplementationAuthority.sol SHA3: e21fb48d1f3e66d4fa4f2a3fb8e46186f85656dd671518de07771cc40358d63b File: ./contracts/proxy/authority/TREXImplementationAuthority.sol SHA3: a4d5db4102684e5b5ab221721e82c19dc7aeec1c6e7dfa4290b6c07b645c7804 File: ./contracts/proxy/ClaimTopicsRegistryProxy.sol

	SHA3: ea04e701b4ab007715801ddce12500c9db4627f57216f811cb3918430ff66c0a File: ./contracts/proxy/IdentityRegistryProxy.sol SHA3: a98b318ff2355e036a9ec21f73090f9d650a214ba82ccc9bc08c1c14babfef47 File: ./contracts/proxy/IdentityRegistryStorageProxy.sol SHA3: 923060438b1c8f23349dfeb0ec0374be790ac7821413f801d5d0bf23e9730b81 File: ./contracts/proxy/ModularComplianceProxy.sol SHA3: 124b5cbfc6ee14905dfdbde1d35f8b46f390e98710fd1ad55d42772ba7ced1d6 File: ./contracts/proxy/TokenProxy.sol SHA3: 5352556e17a4b50e2575c9422ef6f8218fe3eaebc067dde59ff26d767e5b3000 File: ./contracts/proxy/TrustedIssuersRegistryProxy.sol SHA3: 8928506d053358ce770d15ccc7f7f8d05bb39bebacf031d32b6fa0dcb006045f File: ./contracts/registry/implementation/ClaimTopicsRegistry.sol SHA3: 744041b925d6858178599ce3931878aa8b7de6ab669324f93441d09388bace57 File: ./contracts/registry/implementation/IdentityRegistry.sol SHA3: 1ef4bca6f0f6f94d031a3f61d019895d8fd2b9673f003833cb3d0620414809fb File: ./contracts/registry/implementation/IdentityRegistryStorage.sol SHA3: 28627f2dfb4c1ec07ffc37967fd298657e60437d7efc319e55140e83107659e1 File: ./contracts/registry/implementation/TrustedIssuersRegistry.sol SHA3: fdf95fee9e771fad5b562d975fae14d1d9ed7b1314ece0668d3839596f25d3da File: ./contracts/registry/interface/IClaimTopicsRegistry.sol SHA3: fde4fec65bb00c06d2ff2edb1590c7040165a00088adb18feb3cb674519f4c2e File: ./contracts/registry/interface/IIdentityRegistry.sol SHA3: 79ea2359dea51b59d7dfd5e29d506e7f5a6ad79b9029c277514e91daf8a64999 File: ./contracts/registry/interface/IIdentityRegistryStorage.sol SHA3: 6155496dea2aaf5728f31556e4991bacff8c133afd34164345a26050b2dca0a4 File: ./contracts/registry/interface/ITrustedIssuersRegistry.sol SHA3: 7e59af08692098ec3df216ae6e0beea1f2da507a51c3e4d55dd9acdb96bdb94e File: ./contracts/registry/storage/CTRStorage.sol SHA3: 78ab672808d102b5d6fb180adc65958ae7ba4e67458f26e3017bb96db1799505 File: ./contracts/registry/storage/IRSStorage.sol SHA3: 5877167801d60d1a00c0fd0884dfb6f618bb79659884f3d865df485c03aee2e2 File: ./contracts/registry/storage/IRStorage.sol SHA3: ec07dbec0113220db6294d922d7b8fc4c8eda8ce86c5a3b1d872bc168b4811c4 File: ./contracts/registry/storage/TIRStorage.sol SHA3: 81f037d7af5743bc9b5f61963019c254645ac1c5dbfbd731d2283d8bc52b18d0 File: ./contracts/roles/AgentRole.sol SHA3: 1810a8b9c18f2563e08343dacc8a39b4bd2062def2bec63b8fec95031ef0fad4 File: ./contracts/roles/AgentRoleUpgradeable.sol

	SHA3: d90e9e0a00e918ad79e8a0bc26121142d7498d03a1f39977722acefe1a1b8b8b File: ./contracts/roles/Roles.sol SHA3: 5d4df60d9b6f87cf25344d5ecc9402e54e5bdc00d43b1039a751fbc031eeca89 File: ./contracts/token/IToken.sol SHA3: 66438cde83c9b5cc734af1930b08e81c6d05a9a95ef13dd4df332aca429ecf55 File: ./contracts/token/Token.sol SHA3: 3e429e3182e6a36077617ac18716c8af5c2bd3c819598bc4d2dc0d9523eebd7f File: ./contracts/token/TokenStorage.sol SHA3: e682343786e26a5d2aec7e459a3530763683aa2d0e6bc913efa62eb53bcb7f9d

Second review scope

Repository	https://github.com/TokenySolutions/T-REX/tree/hacken-corrections
Commit	3491272c566bfc0de37411858f9780e098b162bf
Contracts Addresses	Not Deployed
Contracts	File: ./contracts/compliance/modular/IModularCompliance.sol SHA3: d7f72c7301df7885e89a0b31160b4de090c7498387231066c33662d238ff9681 File: ./contracts/compliance/modular/MCStorage.sol SHA3: ef8853baaef156f8dabce10f7458726395122a691728dfe72a7f869289fafe25 File: ./contracts/compliance/modular/ModularCompliance.sol SHA3: 342d86bd04f628b8cafe72c0891b534f6e0f185938e200730e7cff10c0fc3eec File: ./contracts/compliance/modular/modules/AbstractModule.sol SHA3: 0003ace97ec9f74cb15eb1afab69e90820aed9eb6624d026ab5eaec93438da88 File: ./contracts/compliance/modular/modules/CountryRestrictModule.sol SHA3: ea87e647c4f458a1032d8c717915056a16c7e077f2f4c1c5a46190759a111f0e File: ./contracts/compliance/modular/modules/IModule.sol SHA3: 3a2a3a692c531a0384d279ecaa562f3090116bd9df2a7df80ffe2b97035208ad File: ./contracts/factory/ITREXFactory.sol SHA3: ec429a1a4b6578e7b240fa780c88253cf81d9bfde25f456f9785e8d3647605c0 File: ./contracts/factory/TREXFactory.sol SHA3: c945f1c336ea18bb799544aa12143f5ff2a3dcb08db29f659a682afae1466de1 File: ./contracts/proxy/AbstractProxy.sol SHA3: 58462e5792364d328748975d254d13837254687d987d8099bfed3140f858e7ff File: ./contracts/proxy/authority/IAFactory.sol SHA3: 8d07ca2d7193c8fb449584a4d22f58769f87433e42f80a57754deda33bc7c2d8 File: ./contracts/proxy/authority/IIAFactory.sol SHA3: bde7975653016fd044f2e8543334acca2f080daf1fd7c16f8c7ec237e6aa527b File: ./contracts/proxy/authority/ITREXImplementationAuthority.sol SHA3: cf9af87afc36645b8d711d2e539c38639ffe2c8dba220d244406bc6c9f253261 File: ./contracts/proxy/authority/TREXImplementationAuthority.sol SHA3: b36e911d7a054347d6fded0e0c31188dacaa0234293073f4087b1cf30a73de2a File: ./contracts/proxy/ClaimTopicsRegistryProxy.sol

	SHA3: 42e8ab642fe28389010b40a3680e218797803d99fc32d972db55967f0010ba76 File: ./contracts/proxy/IdentityRegistryProxy.sol SHA3: b84f442e5380d84234f7976dbf6b39145b57d4b6a835f72cbe09b0c9ffad83f4 File: ./contracts/proxy/IdentityRegistryStorageProxy.sol SHA3: 153c6ca10911ecbeaffe36eab8fe0f08e7753709691762e6713111c5ce884908 File: ./contracts/proxy/interface/IProxy.sol SHA3: 4162cea8f46c08837f9141c44c5aa28897482ca67e8332856543ea9f2945ba43 File: ./contracts/proxy/ModularComplianceProxy.sol SHA3: 651227a0687bb1c26df279a7b6f8ef6def661c78e2d6135f27414d83a2ce0438 File: ./contracts/proxy/TokenProxy.sol SHA3: 97b9b26e52b2a130df9072cd17b4ef2845dedf6f6b575ddbde2a58efe138362e File: ./contracts/proxy/TrustedIssuersRegistryProxy.sol SHA3: 1f1be78df772a0df8b03c2f4e2d138d825efe5da2afe20bea067322eafe68fbe File: ./contracts/registry/implementation/ClaimTopicsRegistry.sol SHA3: a473480a122d02bf7aad923844e9942b31e1db07fe6b0775b0fa0870cbce30f7 File: ./contracts/registry/implementation/IdentityRegistry.sol SHA3: 3ae6274aa9ff6d9fc2ed7d855b707d97c237263ec96ca19ca57bb655cd7f6c82 File: ./contracts/registry/implementation/IdentityRegistryStorage.sol SHA3: f6c839ee853fd6fc0dbae9e0840d6a81439b00aef6b01d1a944c553de75f7ea6 File: ./contracts/registry/implementation/TrustedIssuersRegistry.sol SHA3: ed2033bf75681776302f8a4b3aa03de5a1d4c731e68888376e6f1d34a10115e6 File: ./contracts/registry/interface/IClaimTopicsRegistry.sol SHA3: 66358e9ae51db5f4c3045b65cca496fa0be36fd47e9780c33a516829d19cb9d3 File: ./contracts/registry/interface/IIdentityRegistry.sol SHA3: d4004e445a827ab746f416369c62c79d5eb0d491e0ef4e673facea79a2a1ba2a File: ./contracts/registry/interface/IIdentityRegistryStorage.sol SHA3: 797971a9ab8f56f3fbddf638cf274aa6c53e58a8acca4906b070f2fbeacf8345 File: ./contracts/registry/interface/ITrustedIssuersRegistry.sol SHA3: 7a39d1e73c6774d7e430955c757e75659dd711808325677aea864ec9e035a5ea File: ./contracts/registry/storage/CTRStorage.sol SHA3: 4d5b265c637542931ee88891a7bddaffb3b91ccd743e9b4913759802115f2675 File: ./contracts/registry/storage/IRSStorage.sol

	SHA3: e0e5ae5a5330874554bb1f81d8e8b61e2ea2e2aec37f81196433bfece2f7b437 File: ./contracts/registry/storage/IRStorage.sol SHA3: e17931cd92fa76327d4b04370e6418475f39dbf7a4b0cf2fa4e774ba7a27d504 File: ./contracts/registry/storage/TIRStorage.sol SHA3: d90ae384e7f3290e32c845aa5c1824eab54a0f3608e0d0ffc5b06649d851fdd5 File: ./contracts/roles/AgentRole.sol SHA3: a573c0c8f1a0758fcc38feb522eedf27f4cbfe683dfecf1cfd15011a32e5abbe File: ./contracts/roles/AgentRoleUpgradeable.sol SHA3: abd34390d79e49318a021b902665a25b06b554468b0a737bdc66a7b3924cc8e2 File: ./contracts/roles/Roles.sol SHA3: 711d40dbd3b1f82619eb6a4f0175bda877d87682208057b72c9a98648547eb88 File: ./contracts/token/IToken.sol SHA3: 426e369754a989a9d828914687dacb75c1971ce3c24dc9715860188544b5966c File: ./contracts/token/Token.sol SHA3: 72f41033f902b74dca3b0a27868489614c868a1f489993b3e35a3cd767a88317 File: ./contracts/token/TokenStorage.sol SHA3: fbc4b62c44694e61dcc3d0c7af34924b5d531b94371488290b0da79e7149ab3c

Third review scope

Repository	https://github.com/TokenySolutions/T-REX/tree/hacken-corrections
Commit	e32436b93927c5efb4454e8baf14d904f7f14c45
Contracts Addresses	Not Deployed
Contracts	File: ./contracts/compliance/modular/IModularCompliance.sol SHA3: d7f72c7301df7885e89a0b31160b4de090c7498387231066c33662d238ff9681 File: ./contracts/compliance/modular/MCStorage.sol SHA3: ef8853baaef156f8dabce10f7458726395122a691728dfe72a7f869289fafe25 File: ./contracts/compliance/modular/ModularCompliance.sol SHA3: 342d86bd04f628b8cafe72c0891b534f6e0f185938e200730e7cff10c0fc3eec File: ./contracts/compliance/modular/modules/AbstractModule.sol SHA3: 0003ace97ec9f74cb15eb1afab69e90820aed9eb6624d026ab5eaec93438da88 File: ./contracts/compliance/modular/modules/CountryRestrictModule.sol SHA3: ea87e647c4f458a1032d8c717915056a16c7e077f2f4c1c5a46190759a111f0e File: ./contracts/compliance/modular/modules/IModule.sol SHA3: 3a2a3a692c531a0384d279ecaa562f3090116bd9df2a7df80ffe2b97035208ad File: ./contracts/factory/ITREXFactory.sol SHA3: ec429a1a4b6578e7b240fa780c88253cf81d9bfde25f456f9785e8d3647605c0 File: ./contracts/factory/TREXFactory.sol SHA3: c945f1c336ea18bb799544aa12143f5ff2a3dcb08db29f659a682afae1466de1 File: ./contracts/proxy/AbstractProxy.sol SHA3: 973baed4ec326c3b5df252dbcbc0f6f19f6eadc1f02750de9d1505dcb210e93d File: ./contracts/proxy/authority/IAFactory.sol SHA3: 8d07ca2d7193c8fb449584a4d22f58769f87433e42f80a57754deda33bc7c2d8 File: ./contracts/proxy/authority/IIAFactory.sol SHA3: bde7975653016fd044f2e8543334acca2f080daf1fd7c16f8c7ec237e6aa527b File: ./contracts/proxy/authority/ITREXImplementationAuthority.sol SHA3: cf9af87afc36645b8d711d2e539c38639ffe2c8dba220d244406bc6c9f253261 File: ./contracts/proxy/authority/TREXImplementationAuthority.sol SHA3: b36e911d7a054347d6fded0e0c31188dacaa0234293073f4087b1cf30a73de2a File: ./contracts/proxy/ClaimTopicsRegistryProxy.sol

	SHA3: ebe85cc6f3c44fea18b61f7d3d76b61afb51ce8cef24320c2135f48c9f3f49a2 File: ./contracts/proxy/IdentityRegistryProxy.sol SHA3: 8eb4c24ecf9262fa91c62c854aae615587b1fc39a7294ec88a0d2495e7262449 File: ./contracts/proxy/IdentityRegistryStorageProxy.sol SHA3: dc312224f2855bea1c642f88d3aafbcb649c1ae49e33d8984b433ed9f4689ea6 File: ./contracts/proxy/interface/IProxy.sol SHA3: 4162cea8f46c08837f9141c44c5aa28897482ca67e8332856543ea9f2945ba43 File: ./contracts/proxy/ModularComplianceProxy.sol SHA3: 6c0d643d02743911184c1ab42aebd15a56157c01be239b9d2a23e79803e8d49e File: ./contracts/proxy/TokenProxy.sol SHA3: f793ce2a1e312ffbd7f15905edc424dc85fd6753ad858dcb4a1ce46084349760 File: ./contracts/proxy/TrustedIssuersRegistryProxy.sol SHA3: 1d04c2d027c051ef69b782350f3a59e57fa2d9f2a2fba5abaae757a48d3888d9 File: ./contracts/registry/implementation/ClaimTopicsRegistry.sol SHA3: a473480a122d02bf7aad923844e9942b31e1db07fe6b0775b0fa0870cbce30f7 File: ./contracts/registry/implementation/IdentityRegistry.sol SHA3: 3ae6274aa9ff6d9fc2ed7d855b707d97c237263ec96ca19ca57bb655cd7f6c82 File: ./contracts/registry/implementation/IdentityRegistryStorage.sol SHA3: f6c839ee853fd6fc0dbae9e0840d6a81439b00aef6b01d1a944c553de75f7ea6 File: ./contracts/registry/implementation/TrustedIssuersRegistry.sol SHA3: ed2033bf75681776302f8a4b3aa03de5a1d4c731e68888376e6f1d34a10115e6 File: ./contracts/registry/interface/IClaimTopicsRegistry.sol SHA3: 66358e9ae51db5f4c3045b65cca496fa0be36fd47e9780c33a516829d19cb9d3 File: ./contracts/registry/interface/IIdentityRegistry.sol SHA3: d4004e445a827ab746f416369c62c79d5eb0d491e0ef4e673facea79a2a1ba2a File: ./contracts/registry/interface/IIdentityRegistryStorage.sol SHA3: 797971a9ab8f56f3fbddf638cf274aa6c53e58a8acca4906b070f2fbeacf8345 File: ./contracts/registry/interface/ITrustedIssuersRegistry.sol SHA3: 7a39d1e73c6774d7e430955c757e75659dd711808325677aea864ec9e035a5ea File: ./contracts/registry/storage/CTRStorage.sol SHA3: 4d5b265c637542931ee88891a7bddaffb3b91ccd743e9b4913759802115f2675 File: ./contracts/registry/storage/IRSStorage.sol

	SHA3: e0e5ae5a5330874554bb1f81d8e8b61e2ea2e2aec37f81196433bfece2f7b437 File: ./contracts/registry/storage/IRStorage.sol SHA3: e17931cd92fa76327d4b04370e6418475f39dbf7a4b0cf2fa4e774ba7a27d504 File: ./contracts/registry/storage/TIRStorage.sol SHA3: d90ae384e7f3290e32c845aa5c1824eab54a0f3608e0d0ffc5b06649d851fdd5 File: ./contracts/roles/AgentRole.sol SHA3: a573c0c8f1a0758fcc38feb522eedf27f4cbfe683dfecf1cfd15011a32e5abbe File: ./contracts/roles/AgentRoleUpgradeable.sol SHA3: abd34390d79e49318a021b902665a25b06b554468b0a737bdc66a7b3924cc8e2 File: ./contracts/roles/Roles.sol SHA3: 711d40dbd3b1f82619eb6a4f0175bda877d87682208057b72c9a98648547eb88 File: ./contracts/token/IToken.sol SHA3: 426e369754a989a9d828914687dacb75c1971ce3c24dc9715860188544b5966c File: ./contracts/token/Token.sol SHA3: fc091084f57434f87619c1550fffefed84aed52fa76d6c50fee67deec71e1daa File: ./contracts/token/TokenStorage.sol SHA3: fbc4b62c44694e61dcc3d0c7af34924b5d531b94371488290b0da79e7149ab3c

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation by external or internal actors.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation by external or internal actors.
Medium	Medium vulnerabilities are usually limited to state manipulations but cannot lead to assets loss. Major deviations from best practices are also in this category.
Low	Low vulnerabilities are related to outdated and unused code or minor Gas optimization. These issues won't have a significant impact on code execution but affect the code quality

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurement details can be found in the corresponding section of the scoring methodology.

Documentation quality

The total Documentation Quality score is **8** out of **10**.

- White paper is provided.
- Functional requirements are clearly defined.
- Technical description is provided, but not for all contracts (i.e. TREXFactory and proxy approach).
- NatSpec implementation in Smart Contracts.

Code quality

The total Code Quality score is **9** out of **10**.

- The code duplicates commonly known contracts instead of reusing them (ERC20).
- The development environment is configured.

Test coverage

Test coverage of the project is **83.64%** (branch coverage).

- Deployment and basic user interactions are covered with tests.
- Negative cases coverage is present.
- Interactions by several users are not tested thoroughly.

Security score

As a result of the audit, the code contains **no** issues. The security score is **10** out of **10**.

All found issues are displayed in the “Findings” section.

Summary

According to the assessment, the Customer's smart contract has the following score: **9**.

Table. The distribution of issues during the audit

Review date	Low	Medium	High	Critical
06 December 2022	12	4	4	2
30 December 2022	0	1	1	0

30 March 2023	0	0	0	0

Checked Items

We have audited the Customers' smart contracts for commonly known and more specific vulnerabilities. Here are some items considered:

Item	Type	Description	Status
Default Visibility	SWC-100 SWC-108	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	Passed
Integer Overflow and Underflow	SWC-101	If unchecked math is used, all math operations should be safe from overflows and underflows.	Passed
Outdated Compiler Version	SWC-102	It is recommended to use a recent version of the Solidity compiler.	Passed
Floating Pragma	SWC-103	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	Passed
Unchecked Call Return Value	SWC-104	The return value of a message call should be checked.	Passed
Access Control & Authorization	CWE-284	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	Passed
SELFDESTRUCT Instruction	SWC-106	The contract should not be self-destructible while it has funds belonging to users.	Not Relevant
Check-Effect-Interaction	SWC-107	Check-Effect-Interaction pattern should be followed if the code performs ANY external call.	Passed
Assert Violation	SWC-110	Properly functioning code should never reach a failing assert statement.	Passed
Deprecated Solidity Functions	SWC-111	Deprecated built-in functions should never be used.	Passed
Delegatecall to Untrusted Callee	SWC-112	Delegatecalls should only be allowed to trusted addresses.	Passed
DoS (Denial of Service)	SWC-113 SWC-128	Execution of the code should never be blocked by a specific contract state unless required.	Passed

Race Conditions	SWC-114	Race Conditions and Transactions Order Dependency should not be possible.	Passed
Authorization through tx.origin	SWC-115	tx.origin should not be used for authorization.	Passed
Block values as a proxy for time	SWC-116	Block numbers should not be used for time calculations.	Not Relevant
Signature Unique Id	SWC-117 SWC-121 SWC-122 EIP-155 EIP-712	Signed messages should always have a unique id. A transaction hash should not be used as a unique id. Chain identifiers should always be used. All parameters from the signature should be used in signer recovery. EIP-712 should be followed during a signer verification.	Not Relevant
Shadowing State Variable	SWC-119	State variables should not be shadowed.	Passed
Weak Sources of Randomness	SWC-120	Random values should never be generated from Chain Attributes or be predictable.	Not Relevant
Incorrect Inheritance Order	SWC-125	When inheriting multiple contracts, especially if they have identical functions, a developer should carefully specify inheritance in the correct order.	Passed
Calls Only to Trusted Addresses	EEA-Level-2 SWC-126	All external calls should be performed only to trusted addresses.	Passed
Presence of unused variables	SWC-131	The code should not contain unused variables if this is not justified by design.	Passed
EIP standards violation	EIP	EIP standards should not be violated.	Passed
Assets integrity	**Custom**	Funds are protected and cannot be withdrawn without proper permissions.	Passed
User Balances manipulation	**Custom**	Contract owners or any other third party should not be able to access funds belonging to users.	Passed
Data Consistency	**Custom**	Smart contract data should be consistent all over the data flow.	Passed
Flashloan Attack	**Custom**	When working with exchange rates, they should be received from a trusted source and not be vulnerable to short-term rate changes that can be achieved by using flash loans. Oracles should be used.	Not Relevant

Token Supply manipulation	**Custom**	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the Customer.	Passed
Gas Limit and Loops	**Custom**	Transaction execution costs should not depend dramatically on the amount of data stored on the contract. There should not be any cases when execution fails due to the block Gas limit.	Passed
Style guide violation	**Custom**	Style guides and best practices should be followed.	Passed
Requirements Compliance	**Custom**	The code should be compliant with the requirements provided by the Customer.	Passed
Environment Consistency	**Custom**	The project should contain a configured development environment with a comprehensive description of how to compile, build and deploy the code.	Passed
Secure Oracles Usage	**Custom**	The code should have the ability to pause specific data feeds that it relies on. This should be done to protect a contract from compromised oracles.	Not Relevant
Tests Coverage	**Custom**	The code should be covered with unit tests. Test coverage should be 100%, with both negative and positive cases covered. Usage of contracts by multiple users should be tested.	Failed
Stable Imports	**Custom**	The code should not reference draft contracts, which may be changed in the future.	Passed

System Overview

TREX is an infrastructure to manage *ERC20*-based permissioned tokens by means of a decentralized validator. It is based on 4 pillars:

- ONCHAINID: external on-chain identity management system.
- **Validation Certificates**: referred to as *claims*. These are *ERC734* and *ERC735* based certificates, emitted by trusted third parties and signed on-chain. Each of them is linked to a single *ONCHAINID*.
- **Compliance Rules**: set of rules for a particular token, according to the issuer's requirements and any additional feature. These are checked and managed by a transfer manager role.
- **Eligibility Verification System (EVS)**: validator that filters the P2P transactions of permissioned tokens, according to the *claims* each peer holds and the compliance rules of the tokens. These rules are linked to the identities of transaction receivers and the global distribution of tokens at a certain time.

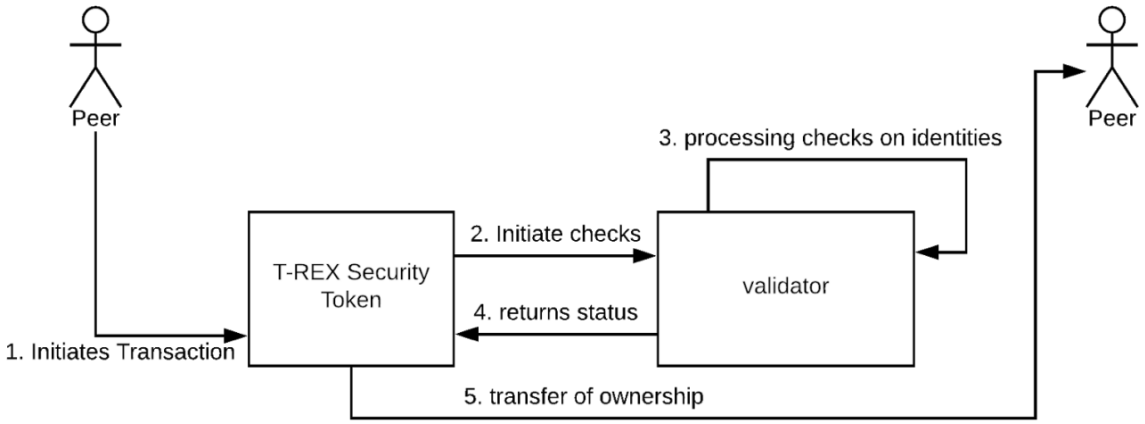

Figure 3 : illustration of a T-REX permissioned token transaction

*from the TREX Whitepaper

Privileged roles

- The owner and the agent roles have huge control over the contract's properties, including pausing, unpausing, minting, burning, forced transfers, adding/removing agents/compliances, changing the implementation, and so on.

Risks

- The repository contains **out-of-scope** contracts, the secureness and reliability of which could not be verified by the current audit.

Findings

■■■■ Critical

1. Access Control Violation / Denial of Service

The *bindCompliance(address _compliance)* function in the AbstractModule.sol contract is available for calling by everyone as long as they give their own address as input. This results in the *complianceBound* mapping variable to be updated so that the new given address returns true. This means that the attacker can now call the *unbindCompliance()* function, bypassing the *onlyComplianceCall* modifier and unbind any compliance contract from the module.

This results in the *mint()* function of Token.sol contract to give DoS(Denial of Service). The error flow is the following:

- *mint()* is called in the Token.sol contract.
- *canTransfer()* is called on the tokenCompliancein the ModularCompliance.sol contract
- *moduleCheck()* is called for every module in the compliance
- *onlyBoundCompliance* modifier will revert because even though the module is checked in the for loop in *canTransfer()* function of ModularCompliance.sol contract, there will be no such compliance stored in the target module.

Paths:

./contracts/token/Token.sol : mint()

./contracts/compliance/modular/ModularCompliance.sol : canTransfer()

./contracts/compliance/modular/modules/AbstractModule.sol : bindCompliance(), unbindCompliance()

./contracts/compliance/modular/modules/CountryRestrictModule.sol : moduleCheck()

Recommendation: as stated in the doc comments for *bindCompliance()* function, it should only be callable by compliance contracts themselves and no other address or contract. A check for this should be added. Architectural changes, such as defining roles in the module contracts, might be considered.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

2. Access Control Violation

In IdentityRegistryStorage.sol, two critical functions *bindIdentityRegistry* and *unbindIdentityRegistry* have no restricted access. However, their corresponding NatSpec defines that these functions can only be called by the IdentityRegistryStorage contract owner.

In Token.sol, *transferFrom()* has no restricted access, although it is a critical function.

Path:

./contracts/registry/implementation/IdentityRegistryStorage.sol:

- bindIdentityRegistry()
- unbindIdentityRegistry()

Recommendation: add an access control mechanism to those functions.

Status: Fixed

■■■ High

1. **Requirements Violation / Missing Validation**

   In Token.sol, the function *burn* does not check the amount to burn is smaller than the balance of the user (both free and frozen tokens) as it is specified in the NatSpec of that function.

   Path:

   ./contracts/token/Token.sol: burn().

   Recommendation: add a *require* to prevent the amount to burn is not exceeding the user's token balance.

   Status: Fixed

   (Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

2. **Non-Finalized Code**

   In *transferred()*, the call *IModule().moduleTransferAction()* has no function body in most modules. The expected behavior is then not clear or seems not to be finalized. It is defined in ConditionalTransferModule.sol, which is out of the audit scope.

   In *created()*, the call *IModule().moduleMintAction()* has no function body in any module. The expected behavior is then not clear or seems not to be finalized.

   In *destroyed()*, the call *IModule().moduleBurnAction()* has no function body in any module. The expected behavior is then not clear or seems not to be finalized.

   Path:

   ./ModularCompliance.sol: transferred(), created(), destroyed().

   Recommendation: define the function body of the in-scope contracts or define their behavior if this is intended.

   Status: Fixed

3. **Unverifiable Logic**

The module *onchain-id*, integrated in several contracts, is out of the audit scope, and therefore its behavior cannot be considered safe.

Paths:

.contracts/registry/implementation/IdentityRegistry.sol:
- identity()
- isVerified()
- registerIdentity()
- batchRegisterIdentity()
- updateIdentity()
- deleteIdentity()

.contracts/registry/implementation/IdentityRegistryStorage.sol:
- storedIdentity()
- addIdentityToStorage()
- modifyStoredIdentity()
- removeIdentityStorage().

.contracts/registry/implementation/TrustedIssuersRegistry.sol:
- addTrustedIssuer()
- removeTrustedIssuer()
- updateIssuerClaimTopics()
- getTrustedIssuers()
- getTrustedIssuerClaimTopics().

.contracts/registry/storage/IRSStorage.sol.
.contracts/registry/storage/TIRStorage.sol.
.contracts/registry/interface/IIdentityRegistry.sol.
.contracts/registry/interface/IdentityRegistryStorage.sol.
.contracts/registry/interface/ITrustedIssuersRegistry.sol.
.contracts/token/Token.sol: recoveryAddress().

Recommendation: define the function body of the in-scope contracts or define their behavior if this is intended.

Status: Fixed

(Revised commit: e32436b93927c5efb4454e8baf14d904f7f14c45)

4. Standard Violation

The contract inherits *AgentRoleUpgradeable* and *TokenStorage,* but neither is designed to support upgradability. Though the *AgentRoleUpgradeable* contract inherits an upgradable contract, it does not have “gaps” to support the future addition of new fields.

In a case when new variables will be added, the storage will be mixed.

The same situation is with other contracts that potentially can be upgraded since they are used through proxies.

Paths: Token.sol

Recommendation: add “gaps” into parent contracts to allow addition of new variables in future upgrades. Follow the upgradability best practices.

Status: Mitigated (with Customer notice)(Since AgentRoleUpgradeable is already deployed, changes cannot be made. However, the provided recommendation was applied for storage contracts) | Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

Medium

1. Denial of Service (DoS)

In various parts of the contracts, for loops that depend on variable array lengths are used. These arrays are not capped in length and can get as large as privileged roles want.

In TREXFactory, when calling deployTREXSuite, the block Gas limit can be overcome since it is performing so many costly operations related to contract deployments.

This may result in Denial of Service in the corresponding functions if the arrays get too large to the point that Gas needed to execute and iterate through the arrays get larger than the maximum Gas the EVM allows.

Paths:

- ./contracts/compliance/modular/ModularCompliance.sol
  - removeModule()
  - canTransfer()
  - destroyed()
  - created()
  - transferred()
- ./contracts/compliance/modular/modules/CountryRestrictModule.so
  - batchRestrictCountries()
  - batchUnrestrictCountries()
- ./contracts/factory/TREXFactory.sol
  - deployTREXSuite() : lines 165, 168, 173, 176, 179
- ./contracts/registry/implementation/ClaimTopicsRegistry.sol
  - addClaimTopics()
  - removeClaimTopics()
- ./contracts/registry/implementation/IdentityRegistryStorage.sol
  - unbindIdentityRegistry()
- ./contracts/registry/implementation/TrustedIssuersRegistry.sol
  - removeTrustedIssuer()
  - isTrustedIssuer()
  - hasClaimTopics()
- ./contracts/registry/implementation/IdentityRegistry.sol
  - isVerified() : Lines 204, 209

Recommendation:

the variable arrays that the for loops depend on could be given hard caps so that they cannot be larger than some value, which would ensure there are no Gas limit excesses.

In the case of deployTREXSuite, a modular approach where each contract is deployed by a different function (and thus transaction) can sort a possible DOS.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

2. **Check Effects Interaction Violation**

During the functions execution, some state variables are updated after the external calls, which is against best practices.

This may lead to reentrancies, race conditions, and denial of service vulnerabilities during the implementation of new functionality.

In Token.sol contract;

- In *transferFrom()* function, *_tokenCompliance.transferred()* external function call is done before internal *_transfer()* function.
- In *transfer()* function, *_tokenCompliance.transferred()* external function call is done before internal *_transfer()* function.
- In *forcedTransfer()*, function, *_tokenCompliance.transferred()* external function call is done before internal *_transfer()* function.
- In *mint()*, if *tokenCompliance.created()* suppose any effects for this action (although in this commit, there is no functionality associated with this call), then it is performed before *_mint()* (interactions).
- In *burn()*, *_tokenCompliance.destroyed()* external call is done before the burn of tokens (interactions).
- In the *recoveryAddress()* function the tokenIdentityRegistry.registerIdentity() and tokenIdentityRegistry.deleteIdentity() external calls are made before contract state changes.

In ModularCompliance.sol contract;

- In *addModule()* function, the IModule(_module).bindCompliance() external call is done before state variable changes.
- In *removeModule()* function, the IModule(_module).unbindCompliance() external call is done before state variable changes.

Paths:

- ./contracts/compliance/modular/ModularCompliance.sol
  - addModule()
  - removeModule()

- ./contracts/token/Token.sol
  - transfer()
  - transferFrom()
  - forcedTransfer()
  - mint()
  - burn()
  - recoveryAddress()

Recommendation: common best practices should be followed, functions should be implemented according to the Check-Effect-Interaction pattern. In *transferFrom()* and *transfer(),* call *tokenCompliance.transferred()* after *_transfer()*. In *forcedTransfer()*, call *tokenCompliance.transferred()* after *_transfer()*. In *mint()*, call *tokenCompliance.created()* after *_mint()*. In *burn()*, call *tokenCompliance.destroyed()* after *_burn()*. Use a similar approach in the rest of the indicated functions.

Status: Fixed

(Revised commit: e32436b93927c5efb4454e8baf14d904f7f14c45)

3. **Unscalable Functionality: Copy of Well-Known Contracts**

In Token.sol, *pausable* functionality is integrated manually instead of using OpenZeppelin library.

In Token.sol, *ERC20* functionality is integrated manually instead of using OpenZeppelin library.

Well-known contracts from projects like OpenZeppelin should be imported directly from the source as the projects are in development and may update the contracts in the future. This can lead to unexpected errors in case of accidental or inattentive modification.

Paths:

./contracts/token/Token.sol: paused(), pause(), unpause(), whenPaused(), whenNotPaused(), transfer(), transferFrom(), init()
./contracts/token/TokenStorage.sol: tokenPaused
./contracts/token/IToken.sol: Paused, Unpaused, paused(), pause(), unpause()

Recommendation: import the contract directly from the source, avoid modifying them.

Status: Mitigated (with Customer notice)(Since the deployment of some contracts has been done, it cannot be changed; however, implementation and storage functions are separated)

4. **Missing Validation**

In CountryRestrictModule.sol, *batchRestrictCountries()* and *batchUnrestrictCountries()* should require whether the country is already restricted or not, as in *addCountryRestriction()* and *removeCountryRestriction()*.

In Token.sol, the function forcedTransfer does not check that *balanceOf(_from)* is bigger or equal to the input *_amount*. This can lead to errors when updating *frozenTokens[_from]* for the case in which *_amount > freeBalance.*

Paths:

./contracts/compliance/modular/modules/CountryRestrictModule.sol:
- batchRestrictCountries()
- batchUnrestrictCountries()

./contracts/token/Token.sol :

- forcedTransfer()

Recommendation: add *require* conditions in the mentioned functions to make sure the code behaves as intended.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

Low

1. Floating Pragma

In every Solidity file in the scope, the expression of *pragma solidity ^0.8.0;* is used while specifying the pragma version.

This may result in the contracts being deployed using the wrong pragma version, which is different from the one they were tested with. For example, they might be deployed using an outdated pragma version which may include bugs that affect the system negatively.

Paths:

./contracts/token/IToken.sol
./contracts/token/Token.sol
./contracts/token/TokenStorage.sol
./contracts/roles/AgentRole.sol
./contracts/roles/AgentRoleUpgradeable.sol
./contracts/roles/Roles.sol
./contracts/registry/implementation/ClaimTopicsRegistry.sol
./contracts/registry/implementation/IdentityRegistry.sol
./contracts/registry/implementation/IdentityRegistryStorage.sol
./contracts/registry/implementation/TrustedIssuersRegistry.sol
./contracts/registry/interface/IClaimTopicsRegistry.sol
./contracts/registry/interface/IIdentityRegistry.sol
./contracts/registry/interface/IIdentityRegistryStorage.sol
./contracts/registry/interface/ITrustedIssuersRegistry.sol
./contracts/registry/storage/CTRStorage.sol
./contracts/registry/storage/IRSStorage.sol
./contracts/registry/storage/IRStorage.sol
./contracts/registry/storage/TIRStorage.sol
./contracts/proxy/authority/ITREXImplementationAuthority.sol
./contracts/proxy/authority/TREXImplementationAuthority.sol
./contracts/proxy/ClaimTopicsRegistryProxy.sol

./contracts/proxy/IdentityRegistryProxy.sol
./contracts/proxy/IdentityRegistryStorageProxy.sol
./contracts/proxy/ModularComplianceProxy.sol
./contracts/proxy/TokenProxy.sol
./contracts/proxy/TrustedIssuersRegistryProxy.sol
./contracts/factory/ITREXFactory.sol
./contracts/factory/TREXFactory.sol
./contracts/compliance/modular/IModularCompliance.sol
./contracts/compliance/modular/MCStorage.sol
./contracts/compliance/modular/ModularCompliance.sol
./contracts/compliance/modular/modules/AbstractModule.sol
./contracts/compliance/modular/modules/CountryRestrictModule.sol
./contracts/compliance/modular/modules/IModule.sol

Recommendation: lock the pragma version and consider known bugs (https://github.com/ethereum/solidity/releases) for the compiler version that is chosen.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

2. **Inefficient Gas Model: SafeMath**

In some cases, SafeMath is unnecessary since over/under-flow is protected by *require* statements and other checks. Performing the operations as *unchecked* will save some Gas.

In Token.sol's *_transfer()* the update of token balances can be calculated as *unchecked*. In *_mint()*, the balances can be calculated as *unchecked* since it cannot exceed *totalSupply* (it will act as an overflow check). In *_burn()*, the update of token balances can be calculated as *unchecked*.

Path:

./contracts/token/Token.sol: _transfer(), _mint(), _burn().

Recommendation: perform the operations under *unchecked*.

Status: Mitigated (with Customer notice)

3. **Inefficient Gas Model: Unnecessary Variable Definition**

In TREXFactory.sol, function *deploy*, a new variable is defined instead of reusing the input parameter *bytecode*, resulting in the expense of unnecessary Gas.

Path:

./contracts/factory/TREXFactory.sol: deploy().

Recommendation: use the input argument *bytecode* directly instead of defining the new variable *implInitCode*.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

4. **Inefficient Gas Model: Function Visibility**

   In ModularCompliance.sol, the visibility getTokenBound() can be set as external.

   In TREXFactory.sol, all deploy functions' visibility can be set as private.

   In TREXImplementationAuthority.sol, all functions can be set as external.

   In ModularCompliance.sol, ClaimTopicsRegistry.sol, IdentityRegistry.sol, IdentityRegistryStorage.sol, TrustedIssuersRegistry.sol and Token.sol the visibility of init() can be set as external.

   Paths:

   ./contracts/compliance/modular/ModularCompliance.sol: init(), getTokenBound().
   ./contracts/factory/TREXFactory.sol: deploy(), deployTIR(), deployCTR(), deployMC(), deployIRS(), deployIR(), deployToken().
   .contracts/proxy/authority/TREXImplementationAuthority.sol: all functions.
   ./contracts/token/Token.sol: init().
   ./contracts/registry/implementation/ClaimTopicsRegistry.sol: init().
   ./contracts/registry/implementation/IdentityRegistry.sol: init().
   ./contracts/registry/implementation/IdentityRegistryStorage.sol: init().
   ./contracts/registry/implementation/TrustedIssuersRegistry.sol: init().

   Recommendation: specify function visibilities according to best practices or document current behavior.

   Status: Mitigated (with Customer notice)

5. **Unused Import**

   The import of AgentRole.sol in the contract AbstractModule.sol is unnecessary. The functionality is not used.

   In Token.sol, the imported interfaces IERC734, IERC735 and IClaimTopicsRegistry are not used.

   Paths:

   ./contracts/compliance/modular/modules/AbstractModule.sol

   ./contracts/token/Token.sol

   Recommendation: unused import should be removed.

   Status: Fixed

   (Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

6. Style Guide: Order of Functions

The provided projects should follow the official guidelines. Functions should be grouped according to their visibility and ordered:

1. Constructor
2. Receive function (if exists)
3. Fallback function (if exists)
4. External
5. Public
6. Internal
7. Private

Paths:
./contracts/compliance/modular/modules/CountryRestrictModule.sol
./contracts/compliance/modular/ModularCompliance.sol
./contracts/factory/TREXFactory.sol
./contracts/registry/implementation/ClaimTopicsRegistry.sol
./contracts/registry/implementation/IdentityRegistry.sol
./contracts/registry/implementation/IdentityRegistryStorage.sol
./contracts/registry/implementation/TrustedIssuersRegistry.sol
./contracts/token/Token.sol

Recommendation: follow the official Solidity guidelines.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

7. Style Guide: Order of Layout

The provided projects should follow the official guidelines. Inside each contract, library or interface, use the following order:

1. Type declarations
2. State variables
3. Events
4. Modifiers
5. Functions

Paths:
./contracts/compliance/modular/modules/CountryRestrictModule.sol
./contracts/factory/ITREXFactory.sol
./contracts/proxy/authority/TREXImplementationAuthority.sol
./contracts/roles/AgentRole.sol
./contracts/roles/AgentRoleUpgradeable.sol
./contracts/token/Token.sol

Recommendation: follow the official Solidity guidelines.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

8. Style Guide: Quotes

The provided projects should follow the official guidelines. Strings should be quoted with double-quotes instead of single-quotes.

Paths:

./contracts/token/IToken.sol
./contracts/token/Token.sol
./contracts/token/TokenStorage.sol
./contracts/roles/AgentRole.sol
./contracts/roles/AgentRoleUpgradeable.sol
./contracts/roles/Roles.sol
./contracts/registry/implementation/ClaimTopicsRegistry.sol
./contracts/registry/implementation/IdentityRegistry.sol
./contracts/registry/implementation/IdentityRegistryStorage.sol
./contracts/registry/implementation/TrustedIssuersRegistry.sol
./contracts/registry/interface/IIdentityRegistry.sol
./contracts/registry/interface/IIdentityRegistryStorage.sol
./contracts/registry/interface/ITrustedIssuersRegistry.sol
./contracts/registry/storage/IRSStorage.sol
./contracts/registry/storage/IRStorage.sol
./contracts/registry/storage/TIRStorage.sol
./contracts/proxy/authority/TREXImplementationAuthority.sol
./contracts/proxy/ClaimTopicsRegistryProxy.sol
./contracts/proxy/IdentityRegistryProxy.sol
./contracts/proxy/IdentityRegistryStorageProxy.sol
./contracts/proxy/ModularComplianceProxy.sol
./contracts/proxy/TokenProxy.sol
./contracts/proxy/TrustedIssuersRegistryProxy.sol
./contracts/factory/TREXFactory.sol
./contracts/compliance/modular/ModularCompliance.sol
./contracts/compliance/modular/modules/AbstractModule.sol
./contracts/compliance/modular/modules/CountryRestrictModule.sol

Recommendation: follow the official Solidity guidelines.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

9. Missing Zero Address/String Input Check

Address parameters are being used without checking against the possibility of *0x0* or empty strings. This can lead to unwanted external calls to *0x0*.

String parameters are being used without checking they are empty.

Paths:
./contracts/token/Token.sol: init(), setOnchainID(), setName(), setSymbol().
./contracts/proxy/TrustedIssuersRegistryProxy.sol: constructor().
./contracts/proxy/TokenProxy.sol: constructor().
./contracts/proxy/IdentityRegistryProxy.sol: constructor().
./contracts/proxy/IdentityRegistryStorageProxy.sol: constructor().
./contracts/proxy/ModularComplianceProxy.sol: constructor().
./contracts/proxy/ClaimTopicsRegistryProxy.sol: constructor().
./contracts/compliance/modular/ModularCompliance.sol: bindToken(), unbindToken(), addModule(), removeModule(), transferred(), created(), destroyed().

./contracts/compliance/modular/modules/AbstractModule.sol: bindCompliance(), unbindCompliance.
./contracts/factory/TREXFactory.sol: setImplementationAuthority().
./contracts/proxy/authority/TREXImplementationAuthority.sol: setTokenImplementation(), setCTRImplementation(), setIRImplementation(), setIRSImplementation(), setTIRImplementation(), setMCImplementation().
./contracts/registry/implementation/IdentityRegistry.sol: init().
./contracts/registry/implementation/IdentityRegistryStorage.sol: addIdentityStorage(), modifyStoredEntity(), modifyStoredInvestorCountry(), removeIdentityFromStorage(), bindIdentityRegistry(), unbindIdentityRegistry().
./contracts/factory/TREXFactory.sol: deployTREXSuite().
./contracts/roles/AfentRole.sol: addAgent(), removeAgent().

Recommendation: zero address check should be done before assigning manual inputs.

Status: **Status**: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

10. NatSpec Typo

There are some errors or misplacements in several contracts that can be easily solved.

In IToken.sol, the NatSpec of *UpdatedTokenInformation* describes this event as being emitted by the token *constructor* instead of the *init* function. In the *_transfer* function, NatSpec refers to *ERC20-_mint*.

In TIRStorage.sol, the NatSpec of *trustedIssuerClaimTopics* is defined as a “mapping between trusted issuer index and its corresponding claim topics” while the key of the mapping is an address.

In IdentityRegistry.sol, the NatSpec of TrustedIssuersRegistrySet refers to *ClaimTopicsRegistry*.

Path:
./contracts/token/IToken.sol: UpdatedTokenInformation, _transfer()

Recommendation: correct the NatSpec descriptions.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

11. Error Message Typo

Some error messages contain small typos, leading to confusions.

Paths:
./contracts/registry/implementation/TrustedIssuersRegistry.sol: removeTrustedIssuer().

./contracts/registry/implementation/IdentityRegistryStorage.sol: addIdentityStorage(), modifyStoredIdentity(), removeIdentityFromStorage().

Recommendation: correct the error messages.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

12. Unindexed Events

In some contracts, no indexed parameters are used in events. If such events are providing relevant enough information to fetch, they should. If this is the intended behavior, this issue should be skipped.

Paths:
./contracts/compliance/modular/modules/CountryRestrictModule.sol.
./contracts/compliance/modular/modules/IModule.sol.
./contracts/proxy/authority/TREXImplementationAuthority.sol.

Recommendation: use indexed events if necessary.

Status: Fixed

(Revised commit: 3491272c566bfc0de37411858f9780e098b162bf)

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date of this report, with cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted to and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, Consultant cannot guarantee the explicit security of the audited smart contracts.