CERTIK

LCX

Security Assessment

November 4th, 2020

For :

LCX AG, Herrengasse 6, 9490 Vaduz Liechtenstein www.LCX.com

RVW Security Token for RVW Limited

Tokenizing the Roe V. Wade Movie

By :

Alex Papageorgiou @ CertiK

alex.papageorgiou@certik.org

Georgios Delkos @ CertiK

georgios.delkos@certik.io

Camden Smallwood @ CertiK

camden.smallwood@certik.org

Overview

Project Summary

Project Name	RVW Token
Description	An ERC-20 and ERC-1404 compatible security token implementation.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. eeb44d3917b00bcfbb71ec9a7184b72bec62241f 2. 7014d3c784cb7eed81127496a54a4aae31fd1c04

Audit Summary

Delivery Date	November 4th, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	3
Timeline	October 27th, 2020 - October 28th, 2020

Vulnerability Summary

Total Issues	19	(17 Resolved, 2 Acknowledged)
Total Critical	0	
Total Major	0	
Total Medium	0	
Total Minor	2	(2 Resolved)
Total Informational	17	(15 Resolved, 2 Acknowledged)

Executive Summary

The audit of the RVW security token identified certain `minor` and `informational` findings, which were relayed to the LCX development team. The LCX team reverted back with an updated commit hash that contained the code changes they made in response to our audit. Out of 2 `minor` findings and 17 `informational` findings, all of the `minor` findings were resolved and all but 2 `informational` findings were resolved. The remaining `informational` findings were determined to not compromise the system and can be implemented at the team's will.

Files In Scope

ID	Contract	Location
ERC	ERC1404.sol	contracts/ERC1404.sol
RVW	RVW.sol	contracts/RVW.sol

Findings

ID	Title	Type	Severity	Resolved
ERC-01	Unconventional Naming Conventions	Coding Style	Informational	✓
ERC-02	User-Defined Getters	Gas Optimization	Informational	✓
ERC-03	`require` Checks w/ No Error Message	Coding Style	Informational	✓
ERC-04	Pull-over-Push Pattern	Logical Issue	Minor	✓
ERC-05	Redundant Casting	Gas Optimization	Informational	✓
ERC-06	Redundant `public` Attribute	Language Specific	Informational	✓
RVW-01	User-Defined Getters	Gas Optimization	Informational	✓
RVW-02	Mutability Specifiers Missing	Gas Optimization	Informational	✓
RVW-04	Redundant Statements	Dead Code	Informational	✓
RVW-05	`require` Checks w/ No Error Message	Coding Style	Informational	✓
RVW-06	Pull-over-Push Pattern	Logical Issue	Minor	✓
RVW-07	Redundant Variable Initialization	Coding Style	Informational	✓
RVW-08	User-Defined Getters	Gas Optimization	Informational	◔
RVW-09	Setter Functions to Toggle Mechanic	Gas Optimization	Informational	◔
RVW-10	`ERC20` Variable Shadowing	Gas Optimization	Informational	✓
RVW-11	Redundant Casting	Coding Style	Informational	✓
RVW-12	Early End of Execution	Logical Issue	Informational	✓
RVW-13	Function Visibility Optimization	Gas Optimization	Informational	✓
RVW-14	Return Variable Utilization	Gas Optimization	Informational	✓

ERC-01: Unconventional Naming Conventions

Type	Severity	Location
Coding Style	Informational	ERC1404.sol L26

Description:

The linked line contains a function definition for checking whether an `address` is locked, however the name of the function is `isLockup` and the name of its input variable is `_address` which are slightly confusing.

Recommendation:

We advise that `isLockup` is renamed to `isLocked` and `_address` is renamed to `account` or `wallet`. The declaration `address _address` is slightly confusing and reserved Solidity keywords should be avoided from being utilized as function input names.

Alleviation:

The linked function was indeed renamed to `isLocked` to increase legibility, alleviating this exhibit.

ERC-02: User-Defined Getters

Type	Severity	Location
Gas Optimization	Informational	ERC1404.sol L38, L48-L51

Description:

The linked variables contain user-defined getter functions that are equivalent to their name barring for an underscore (`_`) prefix / suffix.

Recommendation:

We advise that the linked variables are instead declared as `public` and that they are renamed to their respective getter's name as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

Alleviation:

The team alleviated this exhibit by setting the linked variables as `public` and properly adjusting their name.

ERC-03: `require` Checks w/ No Error Message

Type	Severity	Location
Coding Style	Informational	ERC1404.sol L55, L77

Description:

The linked `require` checks contain no error message that accompanies the condition they evaluate.

Recommendation:

We advise that an error message is explicitly set for those `require` checks to aid in the debugging process of the smart contracts.

Alleviation:

Error messages were properly set for all `require` calls.

ERC-04: Pull-over-Push Pattern

Type	Severity	Location
Logical Issue	Minor	ERC1404.sol L70-L80

Description:

The linked functions conduct a transfer of ownership of the contract without necessarily validating that the `newOwner` address is indeed owned by an EOA or a smart contract.

Recommendation:

In Solidity, pull patterns should be applied instead of push patterns when dealing with sensitive contract variables. In this case, a new `owner` should instead be proposed and another function should be implemented that allows the proposed owner to accept his invitation, thus ensuring that the new owner is capable of interacting with the contract.

Alleviation:

A new pattern was implemented whereby a new owner is suggested via `proposeOwnership` and they subsequently accept via `acceptOwnership` in a pull-pattern.

ERC-05: Redundant Casting

Type	Severity	Location
Gas Optimization	Informational	ERC1404.sol L115, L117

Description:

The `updateChecker` function contains an input variable of type `address` that is casted to a type `IERC1404Checks` interface redundantly.

Recommendation:

We advise that the input variable is set to be of `IERC1404Checks` type and the `require` check of L116 utilizes a not-equal comparison with `IERC1404Checks(0)`.

Alleviation:

The function input as well as `require` comparison were properly updated to utilize `IERC1404Checks`.

ERC-06: Redundant `public` Attribute

Type	Severity	Location
Language Specific	Informational	ERC1404.sol L90-L93, L95-L99

Description:

The linked variables are declared as `public` yet they are utilized internally as status codes and the error messages are retrievable via the `messageForTransferRestriction` function.

Recommendation:

We advise that the `public` attribute is removed greatly decreasing the generated bytecode. These variables should instead be set to either `private` or `internal` depending on their use case.

Alleviation:

The `public` attribute was instead replaced by an `internal` attribute properly denoting their functionality and reducing the contract's generated bytecode.

RVW-01: User-Defined Getters

Type	Severity	Location
Gas Optimization	Informational	RVW.sol L58-L60, L70-L73, L77-L83, L85-L92, L94-L99

Description:

The linked variables contain user-defined getter functions that are equivalent to their name barring for an underscore (`_`) prefix / suffix.

Recommendation:

We advise that the linked variables are instead declared as `public` and that they are renamed to their respective getter's name as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

Alleviation:

The team alleviated this exhibit by setting the linked variables as `public` and properly adjusting their name.

RVW-02: Mutability Specifiers Missing

Type	Severity	Location
Gas Optimization	Informational	RVW.sol L60, L66

Description:

The linked variables are assigned to only once, either during their contract-level declaration or during the `constructor`'s execution.

Recommendation:

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables. Please note that the `immutable` keyword only works in Solidity versions `v0.6.5` and up.

Alleviation:

The `decimals` variable was properly set to be `immutable` and thus reduce the gas cost involved in interacting with it.

RVW-04: Redundant Statements

Type	Severity	Location
Dead Code	Informational	RVW.sol L233-L249

Description:

The linked statements do not affect the functionality of the codebase and appear to be either leftovers from test code or older functionality.

Recommendation:

We advise that they are removed to better prepare the code for production environments.

Alleviation:

The LCX development team has acknowledged this exhibit and the dead code is now in use by the burn public functions

RVW-05: `require` Checks w/ No Error Message

Type	Severity	Location
Coding Style	Informational	RVW.sol L306, L339

Description:

The linked `require` checks contain no error message that accompanies the condition they evaluate.

Recommendation:

We advise that an error message is explicitly set for those `require` checks to aid in the debugging process of the smart contracts.

Alleviation:

Error messages were properly set for all `require` calls.

RVW-06: Pull-over-Push Pattern

Type	Severity	Location
Logical Issue	Minor	RVW.sol L326-L342

Description:

The linked functions conduct a transfer of ownership of the contract without necessarily validating that the `newOwner` address is indeed owned by an EOA or a smart contract.

Recommendation:

In Solidity, pull patterns should be applied instead of push patterns when dealing with sensitive contract variables. In this case, a new `owner` should instead be proposed and another function should be implemented that allows the proposed owner to accept his invitation, thus ensuring that the new owner is capable of interacting with the contract.

Alleviation:

A new pattern was implemented whereby a new owner is suggested via `proposeOwnership` and they subsequently accept via `acceptOwnership` in a pull-pattern.

RVW-07: Redundant Variable Initialization

Type	Severity	Location
Coding Style	Informational	RVW.sol L451-L456

Description:

All variable types within Solidity are initialized to their default "empty" value, which is usually their zeroed out representation. Particularly:

- `uint` / `int`: All `uint` and `int` variable types are initialized at `0`
- `address`: All `address` types are initialized to `address(0)`
- `byte`: All `byte` types are initialized to their `byte(0)` representation
- `bool`: All `bool` types are initialized to `false`
- `ContractType`: All contract types (i.e. for a given `contract ERC20 {}` its contract type is `ERC20`) are initialized to their zeroed out address (i.e. for a given `contract ERC20 {}` its default value is `ERC20(address(0))`)
- `struct`: All `struct` types are initialized with all their members zeroed out according to this table

Recommendation:

We advise that the linked initialization statements are removed from the codebase to increase legibility.

Alleviation:

The linked `constructor` was omitted from the contract thus reducing the generated bytecode and associated deployment cost of the contract.

RVW-08: User-Defined Getters

Type	Severity	Location
Gas Optimization	Informational	RVW.sol L449, L458-L463

Description:

The linked variables contain user-defined getter functions that are equivalent to their name barring for an underscore (`_`) prefix / suffix.

Recommendation:

We advise that the linked variables are instead declared as `public` and that they are renamed to their respective getter's name as compiler-generated getter functions are less prone to error and much more maintainable than manually written ones.

Alleviation:

The LCX development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase for strategic reasons.

RVW-09: Setter Functions to Toggle Mechanic

Type	Severity	Location
Gas Optimization	Informational	RVW.sol L489-L511

Description:

The linked functions set a value literal to the underlying variable `_paused` in two separate function implementations.

Recommendation:

We advise that a toggle or single setter function with an input variable is set to decrease the total bytecode of the contract.

Alleviation:

The LCX development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase for strategic reasons.

RVW-10: `ERC20` Variable Shadowing

Type	Severity	Location
Gas Optimization	Informational	RVW.sol L609-L615

Description:

The linked variables are already declared in the ERC20 interface and as such do not need to be redeclared.

Recommendation:

We advise that they are instead passed as literals to the `ERC20` constructor via L630.

Alleviation:

The LCX development team has resolved the issue entirely.

RVW-11: Redundant Casting

Type	Severity	Location
Coding Style	Informational	RVW.sol L655

Description:

The linked `require` check casts `IERC1404` to an `address` variable before comparing.

Recommendation:

We advise that the comparison is instead set to `restrictedTransfer != IERC1404(0)`.

Alleviation:

The LCX development team has acknowledged this exhibit but decided to not apply its remediation in the current version of the codebase for strategic reasons.

RVW-12: Early End of Execution

Type	Severity	Location
Logical Issue	Informational	RVW.sol L710-L712

Description:

The linked `for` loop can end abruptly early if one of the `_issueTokenAndWhitelist` invocations fail.

Recommendation:

We advise that a `try-catch` clause is utilized as it is supported in Solidity version `0.7.0`, and the loop ends early by returning the index up to which the transactions were successful.

Alleviation:

After discussing with the LCX team, we concluded that they will not apply the recommendation of this exhibit as they will handle it off chain.

RVW-13: Function Visibility Optimization

Type	Severity	Location
Gas Optimization	Informational	RVW.sol L706

Description:

The linked function is declared as `public`, contains array function arguments and is not invoked in any of the contract's contained within the project's scope.

Recommendation:

We advise that the functions' visibility specifiers are set to `public` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

Alleviation:

The LCX development team has completely applied the recommendations.

RVW-14: Return Variable Utilization

Type	Severity	Location
Gas Optimization	Informational	RVW.sol L684

Description:

The linked function declarations contain explicitly named `return` variables that are not utilized within the function's code block.

Recommendation:

We advise that the linked variables are either utilized or omitted from the declaration.

Alleviation:

The LCX development team has completely applied the recommendations.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an in-storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Disclaimer

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.