Dollar Protocol

Security Assessment

September 21, 2020

By :
Camden Smallwood @ CertiK
camden.smallwood@certik.org

Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org

DISCLAIMER

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

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.

SUMMARIES

Project Summary

Project Name	Dollar Protocol
Description	A 2 token protocol comprising of Dollars and Shares. Dollars will be the object of stabilization and Shares will be the instrument to speculate in and govern the network.
Platform	Ethereum; Solidity, Yul
Codebase	GitHub Repository
Commits	1. e63c5a563e422b4bd1c2faf7484f8a395b7ac2a9 2. ad9b4402329c5bc5d523a3523a545cb743c4f9c7 3. 3fb9bc5f2bc9f990beb2978148a201406d15308b 4. f6c742d8930127c54dfb85a7ed8f0ba5806fd151

Audit Summary

Delivery Date	Sep. 21, 2020
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Sep. 14, 2020 - Sep. 18 2020

Vulnerability Summary

Total Issues	38
Total Critical	0
Total Major	1
Total Minor	13
Total Informational	24

FINDINGS

ID	Title	Type	Severity
DOL-01	Incorrect import name	Language Specific	Minor
DOL-02	Unnecessary comparison w/ uint256	Implementation	Informational
DOL-03	State variable name not in mixed case	Naming Conventions	Informational
DOL-04	State variable name not in mixed case	Naming Conventions	Informational
DOL-05	Inefficient comparisons w/ uint256	Implementation	Informational
DOL-06	Unnecessary comparison w/ uint256	Implementation	Informational
DOL-07	Rebase lock when supply delta is zero	Data Flow	Major
DOL-08	Inefficient arithmetic operations	Implementation	Informational
DOL-09	Inefficient comparison w/ uint256	Implementation	Informational
DOL-10	Addresses not verified	Implementation	Minor
DOL-11	Addresses not verified	Implementation	Minor
DOL-12	Addresses not verified	Implementation	Minor
DOL-13	Addresses not verified	Implementation	Minor
DOL-14	Addresses not verified	Implementation	Minor
DOL-15	Unlabeled constant numbers	Implementation	Informational
DOL-16	Unlabeled constant numbers	Implementation	Informational
DOL-17	Unlabeled constant numbers	Implementation	Informational
DOL-18	Comparison to boolean constant	Implementation	Informational

FINDINGS

ID	Title	Type	Severity
DOL-19	Comparison to boolean constant, Unlabeled constant numbers, Inefficient comparison w/ uint256	Implementation, Implementation, Implementation	Informational, Informational, Informational
DOL-20	Inefficient comparison w/ uint256	Implementation	Informational
DOL-21	Inefficient comparison w/ uint256	Implementation	Informational
DOL-22	Comparison to boolean constant	Implementation	Informational
DOL-23	Inefficient comparison w/ uint256	Implementation	Informational
DOL-24	Unlabeled constant numbers	Implementation	Informational
DOL-25	Address not verified, Unnecessary return type	Implementation, Implementation	Minor, Informational
DOL-26	Address not verified, Unnecessary return type	Implementation, Implementation	Minor, Informational
DOL-27	Address not verified	Implementation	Minor
DOL-28	Address not verified	Implementation	Minor
DOL-29	Address not verified	Implementation	Minor
DOL-30	Addresses not verified	Implementation	Minor
DOL-31	Addresses not verified	Implementation	Minor
DOL-32	Addresses not verified	Implementation	Minor

DOL-01: Incorrect import name

Type	Severity	Location
Language Specific	Minor	dollars.sol, L5

Description:

The `dollars.sol` file attempted to import the `ISeigniorageShares.sol` file but had it misspelled.

Recommendation:

We suggested to fix the spelling of the filename in order to point to the correct file.

Alleviation:

The recommendation was applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-02: Unnecessary comparison w/ uint256

Type	Severity	Location
Implementation	Informational	dollars.sol, L61

Description:

The `validDiscount` modifier in the `Dollars` contract had a requirement that performed an unnecessary greater-than-or-equal comparison between a uint256 and a zero constant.

Recommendation:

Since uint256 values cannot be negative, we recommended removing the requirement in order to save on the overall cost of gas.

Alleviation:

The recommendation was applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-03: State variable name not in mixed case

Type	Severity	Location
Naming Conventions	Informational	dollars.sol, L75

Description:

The `Dollars` contract had a state variable named `Shares` which was not in mixed case. We pointed out that Solidity has a naming convention that should be followed.

Recommendation:

We recommended renaming the `Shares` state variable to `shares` in the `Dollars` contract.

Alleviation:

The recommendation was applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-04: State variable name not in mixed case

Type	Severity	Location
Naming Conventions	Informational	dollars.sol, L83

Description:

The `Dollars` contract had a state variable named `DollarPolicy` which was not mixed case. We pointed out that Solidity has a naming convention that should be followed.

Recommendation:

We recommended renaming the `DollarPolicy` state variable to `dollarPolicy` in the `Dollars` contract.

Alleviation:

The recommendation was applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-05: Inefficient comparisons w/ uint256

Type	Severity	Location
Implementation	Informational	dollars.sol, L130-L132

Description:

The `burn` function in the `Dollars` contract had requirements that performed inefficient comparisons between uint256 variables and zero.

Recommendation:

Since uint256 values cannot be below zero, we recommended converting the greater-than comparisons to inequality and remove the burningDiscount comparison in order to save on the overall cost of gas.

Alleviation:

The recommendation was applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-06: Unnecessary comparison w/ uint256

Type	Severity	Location
Implementation	Informational	dollars.sol, L186

Description:

The `setMinimumBonusThreshold` function in the `Dollars` contract had a requirement that performed an unnecessary greater-than-or-equal comparison between a uint256 and a zero constant.

Recommendation:

Since uint256 values cannot be negative, we recommended removing the requirement in order to save on the overall cost of gas.

Alleviation:

The recommendation was applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-07: Rebase lock when supply delta is zero

Type	Severity	Location
Data Flow	Major	dollars.sol, L213

Description:

The `rebase` function in the `Dollars` contract failed to disable the `reEntrancyRebaseMutex` state variable when the `supplyDelta` parameter's value was zero, preventing any transfers from taking place until the `rebase` function was called again with a non-zero `supplyDelta` value.

Recommendation:

We recommended removing the return statement in the `if` block that executed when the `supplyDelta` value was zero and adjoining the following `if` block that executed when the `supplyDelta` value was less than zero by turning it into an `else if`, which allowed the function to execute with the correct behavior and safely disable the `reEntrancyRebaseMutex` state variable before returning the total supply.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-08: Inefficient arithmetic operations

Type	Severity	Location
Implementation	Informational	dollars.sol, L206-L207, L218-L219, L227-L228

Description:

The `rebase` function in the `Dollars` contract performed multiple inefficient arithmetic operations without storing an intermediate result, which was non-optimal and created a higher overall cost of gas.

Recommendation:

We recommended performing each operation one time only and storing the result in a local variable in order to reduce the overall cost of gas for the `rebase` function.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-09: Inefficient comparison w/ uint256

Type	Severity	Location
Implementation	Informational	dollars.sol, L290

Description:

The `updateAccount` modifier in the `Dollars` contract had an inefficient greater-than comparison between a uint256 and a zero constant.

Recommendation:

Since uint256 values cannot be negative, we recommended converting the greater-than comparison into an inequality comparison in order to save on the overall cost of gas.

Alleviation:

The recommendation was applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-10: Addresses not verified

Type	Severity	Location
Implementation	Minor	dollars.sol, L338-L343

Description:

The `transfer` function in the `Dollars` contract did not check if the supplied `to` address parameter and the `msg.sender` global were unique addresses before updating the account information for both addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `Dollars` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `validRecipient` modifier in the signature of the `transfer` function.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-11: Addresses not verified

Type	Severity	Location
Implementation	Minor	dollars.sol, L372-L378

Description:

The `transferFrom` function in the `Dollars` contract did not check if the supplied `from` and `to` address parameters and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `Dollars` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `validRecipient` modifier in the signature of the `transferFrom` function.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-12: Addresses not verified

Type	Severity	Location
Implementation	Minor	dollars.sol, L402-L407

Description:

The `approve` function in the `Dollars` contract did not check if the supplied `spender` address parameter and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended Creating a modifier in the `Dollars` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `validRecipient` modifier in the signature of the `approve` function.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-13: Addresses not verified

Type	Severity	Location
Implementation	Minor	dollars.sol, L421-L425

Description:

The `increaseAllowance` function in the `Dollars` contract did not check if the supplied `spender` address parameter and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `Dollars` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `updateAccount` modifier in the signature of the `increaseAllowance` function.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-14: Addresses not verified

Type	Severity	Location
Implementation	Minor	dollars.sol, L439-L443

Description:

The `decreaseAllowance` function in the `Dollars` contract did not check if the supplied `spender` address parameter and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `Dollars` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `updateAccount` modifier in the signature of the `decreaseAllowance` function.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-15: Unlabeled constant numbers

Type	Severity	Location
Implementation	Informational	dollars.sol, L466-L467

Description:

The `consultBurn` function in the `Dollars` contract contained the usage of unlabeled constant numbers, which made the function difficult to review correctly.

Recommendation:

We recommended creating constant variables for the constant numbers used in order to clarify the intended functionality.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-16: Unlabeled constant numbers

Type	Severity	Location
Implementation	Informational	dollars.sol, L502-L503

Description:

The `_burn` function in the `Dollars` contract contained the usage of unlabeled constant numbers, which made the function difficult to review correctly.

Recommendation:

We recommended creating constant variables for the constant numbers used in order to clarify the intended functionality.

Alleviation:

The recommendations were applied in commit f6c742d8930127c54dfb85a7ed8f0ba5806fd151.

DOL-17: Unlabeled constant numbers

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L75

Description:

The `getUsdSharePrice` function in the `DollarsPolicy` contract contained the usage of unlabeled constant numbers, which makes the function difficult to review correctly.

Recommendation

We recommended creating constant variables for the constant numbers used in order to clarify the intended functionality.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-18: Comparison to boolean constant

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L81

Description:

The `initializeOracles` function in the `DollarsPolicy` contract contained a requirement that performed direct comparison to a boolean constant, which is unnecessary.

Recommendation:

Since booleans can be used directly and do not need to be compare to `true` or `false`, we recommended refactoring the requirement.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-19: Comparison to boolean constant, Unlabeled constant numbers, Inefficient comparison w/ uint256

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L81
Implementation	Informational	dollarsPolicy.sol, L94-L99
Implementation	Informational	dollarsPolicy.sol, L113

Description:

The `rebase` function in the `DollarsPolicy` contract contained a requirement that performed a direct comparison to a boolean constant, contained the usage of unlabeled constant numbers (which made the function difficult to review correctly) and had a requirement that performed an inefficient greater-than comparison between a uint256 and a zero constant.

Recommendation:

Since booleans can be used directly and do not need to be compare to `true` or `false`, we recommended refactoring the requirement. We also recommended creating constant variables for the constant numbers used in order to clarify the intended functionality. Since uint256 values cannot be negative, we recommended converting the greater-than comparison into an inequality check in the requirement in order to save on the overall cost of gas.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-20: Inefficient comparison w/ uint256

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L164

Description:

The `setCpi` function in the `DollarsPolicy` contract contained a requirement that performed an inefficient greater-than comparison between a uint256 and a zero constant.

Recommendation:

Since uint256 values cannot be negative, we recommended converting the greater-than comparison into an inequality check in the requirement in order to save on the overall cost of gas.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-21: Inefficient comparison w/ uint256

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L172

Description:

The `setRebaseLag` function in the `DollarsPolicy` contract contained a requirement that performed an inefficient greater-than comparison between a uint256 and a zero constant.

Recommendation:

Since uint256 values cannot be negative, we recommended converting the greater-than comparison into an inequality check in the requirement in order to save on the overall cost of gas.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-22: Comparison to boolean constant

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L181

Description:

The `initializeOracles` function in the `DollarsPolicy` contract contained a requirement that performed a direct comparison to a boolean constant.

Recommendation:

Since booleans can be used directly and do not need to be compare to `true` or `false`, we recommended refactoring the requirement.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-23: Inefficient comparison w/ uint256

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L227

Description:

The `setRebaseTimingParameters` function in the `DollarsPolicy` contract contained a requirement that performed an inefficient greater-than comparison between a uint256 and a zero constant.

Recommendation:

Since uint256 values cannot be negative, we recommended converting the greater-than comparison into an inequality check in the requirement in order to save on the overall cost of gas.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-24: Unlabeled constant numbers

Type	Severity	Location
Implementation	Informational	dollarsPolicy.sol, L258-L265

Description:

The `getAlgorithmicRebaseLag` function in the `DollarsPolicy` contract contained the usage of unlabeled constant numbers, which made the function difficult to review correctly.

Recommendation:

We recommended creating constant variables for the constant numbers used in order to clarify the intended functionality.

Alleviation:

The recommendations were applied in commit 3fb9bc5f2bc9f990beb2978148a201406d15308b.

DOL-25: Address not verified, Unnecessary return type

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L48-L51
Implementation	Informational	seigniorageShares.sol, L48-L51

Description:

The `setDividendPoints` function in the `SeigniorageShares` contract did not verify that the supplied `who` address was a valid address and always returned true, which was unnecessary.

Recommendation:

We recommended refactoring the `setDividendPoints` function to verify that the supplied `who` address is valid and not return a boolean value.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.

DOL-26: Address not verified, Unnecessary return type

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L53-L61
Implementation	Informational	seigniorageShares.sol, L53-L61

Description:

The `mintShares` function in the `SeigniorageShares` contract did not verify that the supplied `who` address was a valid address and always returned true, which was unnecessary.

Recommendation:

We recommended refactoring the `mintShares` function to verify that the supplied `who` address is valid and not return a boolean value.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.

DOL-27: Address not verified

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L103-L108

Description:

The `initializeDollar` function in the `SeigniorageShares` contract did not verify that the supplied `dollarAddress` parameter was a valid address.

Recommendation:

We recommended refactoring the `initializeDollar` function to require that the supplied `dollarAddress` parameter is a valid address.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.

DOL-28: Addresses not verified

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L135-L140

Description:

The `transfer` function in the `SeigniorageShares` contract did not check if the supplied `to` address parameter and the `msg.sender` global were unique addresses before updating the account information for both addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `SeigniorageShares` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `validRecipient` modifier in the signature of the `transfer` function.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.

DOL-29: Addresses not verified

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L169-L175

Description:

The `transferFrom` function in the `SeigniorageShares` contract did not check if the supplied `from` and `to` address parameters and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `SeigniorageShares` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `validRecipient` modifier in the signature of the `transferFrom` function.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.

DOL-30: Addresses not verified

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L199-L204

Description:

The `approve` function in the `SeigniorageShares` contract did not check if the supplied `spender` address parameter and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `SeigniorageShares` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `validRecipient` modifier in the signature of the `approve` function.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.

DOL-31: Addresses not verified

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L224-L228

Description:

The `increaseAllowance` function in the `SeigniorageShares` contract did not check if the supplied `spender` address parameter and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `SeigniorageShares` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `updateAccount` modifier in the signature of the `increaseAllowance` function.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.

DOL-32: Addresses not verified

Type	Severity	Location
Implementation	Minor	seigniorageShares.sol, L242-L246

Description:

The `decreaseAllowance` function in the `SeigniorageShares` contract did not check if the supplied `spender` address parameter and the `msg.sender` global were unique addresses before updating the account information for those addresses or transferring tokens. This allowed for updating the same address multiple times and transferring tokens to itself, which was inefficient and could lead to token issues in some cases.

Recommendation:

We recommended creating a modifier in the `SeigniorageShares` contract that requires that two supplied addresses are not equal, then placing this modifier before the initial `updateAccount` modifier in the signature of the `decreaseAllowance` function.

Alleviation:

The recommendations were applied in commit ad9b4402329c5bc5d523a3523a545cb743c4f9c7.