Veridise. Auditing Report

Hardening Blockchain Security with Formal Methods

FOR

MANTA NETWORK

Wrapped Mountain Protocol USD

DRAFT

Veridise Inc.

December 29, 2023

DRAFT

► **Prepared For:**

Manta Network
https://manta.network

► **Prepared By:**

Jon Stephens
Benjamin Sepanski

► **Contact Us:** contact@veridise.com

► **Version History:**

Dec. 20, 2023 Initial Draft

© 2023 Veridise Inc. All Rights Reserved.

Contents

DRAFT

 © 2023 Veridise Inc.

DRAFT

1 Executive Summary

On Dec. 18, 2023, Manta Network engaged Veridise to review the security of their Wrapped Mountain Protocol USD. The review covered the Solidity source code of the token implementation which was deployed on Manta Pacific at address `0xbdAd407F77f44F7Da6684B416b1951ECa461FB07` *. Veridise conducted the assessment over 2 person-days, with 2 engineers reviewing code over 1 day. The auditing strategy involved a tool-assisted analysis of the source code performed by Veridise engineers as well as extensive manual auditing.

Code assessment. The Wrapped Mountain Protocol USD developers provided the source code of the Wrapped Mountain Protocol USD contracts for review. The source code is very small, as it simply extends the Optimism† contract `OptimismMintableERC20`‡ with permissioned pause functionality. No external documentation from the developers was provided.

Summary of issues detected. The audit uncovered 1 issue, which was assessed to be of warning-level severity by the Veridise auditors. Specifically, V-WUSD-VUL-001 describes the possibility of a centralization risk leading to a denial-of-service.

Disclaimer. We hope that this report is informative but provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the system is secure in all dimensions. In no event shall Veridise or any of its employees be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the results reported here.

DRAFT

* https://pacific-explorer.manta.network/address/0xbdAd407F77f44F7Da6684B416b1951ECa461FB07
† https://www.optimism.io
‡ https://github.com/ethereum-optimism/optimism/blob/v1.2.0/packages/contracts-bedrock/src/universal/OptimismMintableERC20.sol

 © 2023 Veridise Inc.

DRAFT

2 Project Dashboard

Table 2.1: Application Summary.

Name	Address	Type	Platform
Wrapped Mountain Protocol USD	0xbdAd407F77...	Solidity	Manta Pacific

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
Dec. 18 - Dec. 18, 2023	Manual & Tools	2	2 person-day

Table 2.3: Vulnerability Summary.

Name	Number	Resolved
Critical-Severity Issues	0	0
High-Severity Issues	0	0
Medium-Severity Issues	0	0
Low-Severity Issues	0	0
Warning-Severity Issues	1	0
Informational-Severity Issues	0	0
TOTAL	1	0

Table 2.4: Category Breakdown.

Name	Number
Centralization	1

DRAFT

© 2023 Veridise Inc.

DRAFT

3 Audit Goals and Scope

3.1 Audit Goals

The engagement was scoped to provide a security assessment of Wrapped Mountain Protocol USD's smart contract. In our audit, we sought to answer the following questions:

- Can the contract be manipulated such that tokens can be stolen from users?
- Can a user manipulate tokens that are owned by another user?
- Are there risks associated with centralization?
- Are any minting functions properly guarded by access controls?
- Does the token adhere to the behaviors defined in the ERC20 specification?

3.2 Audit Methodology & Scope

Audit Methodology. To address the questions above, our audit involved a combination of human experts and automated program analysis & testing tools. In particular, we conducted our audit with the aid of the following techniques:

- *Static analysis.* To identify potential common vulnerabilities, we leveraged our custom smart contract analysis tool Vanguard, as well as the open-source tool Slither. These tools are designed to find instances of common smart contract vulnerabilities, such as reentrancy and uninitialized variables.
- *Fuzzing/Property-based Testing.* We also leverage fuzz testing to determine if the protocol may deviate from the expected behavior. To do this, we formalize the desired behavior of the protocol as [V] specifications and then use our fuzzing framework OrCa to determine if a violation of the specification can be found.

Scope. The scope of this audit is limited to the `MantaMintableERC20WithBridgeFlag.sol` contract deployed at address `0xbdAd407F77f44F7Da6684B416b1951ECa461FB07` on the Manta Pacific blockchain.

Methodology. Veridise auditors reviewed the reports of previous audits for Wrapped Mountain Protocol USD, inspected the provided tests, and read the Wrapped Mountain Protocol USD documentation. They then began a manual audit of the code assisted by both static analyzers and automated testing.

3.3 Classification of Vulnerabilities

When Veridise auditors discover a possible security vulnerability, they must estimate its severity by weighing its potential impact against the likelihood that a problem will arise. Table 3.1 shows how our auditors weigh this information to estimate the severity of a given issue.

DRAFT

 © 2023 Veridise Inc.

Table 3.1: Severity Breakdown.

	Somewhat Bad	Bad	Very Bad	Protocol Breaking
Not Likely	Info	Warning	Low	Medium
Likely	Warning	Low	Medium	High
Very Likely	Low	Medium	High	Critical

In this case, we judge the likelihood of a vulnerability as follows in Table 3.2:

Table 3.2: Likelihood Breakdown

Not Likely	A small set of users must make a specific mistake
Likely	Requires a complex series of steps by almost any user(s) - OR - Requires a small set of users to perform an action
Very Likely	Can be easily performed by almost anyone

In addition, we judge the impact of a vulnerability as follows in Table 3.3:

Table 3.3: Impact Breakdown

Somewhat Bad	Inconveniences a small number of users and can be fixed by the user
Bad	Affects a large number of people and can be fixed by the user - OR - Affects a very small number of people and requires aid to fix
Very Bad	Affects a large number of people and requires aid to fix - OR - Disrupts the intended behavior of the protocol for a small group of users through no fault of their own
Protocol Breaking	Disrupts the intended behavior of the protocol for a large group of users through no fault of their own

DRAFT

Vulnerability Report 4

In this section, we describe the vulnerabilities found during our audit. For each issue found, we log the type of the issue, its severity, location in the code base, and its current status (i.e., acknowledged, fixed, etc.). Table 4.1 summarizes the issues discovered:

Table 4.1: Summary of Discovered Vulnerabilities.

ID	Description	Severity	Status
V-WUSD-VUL-00	Centralization Risk	Warning	Open

DRAFT

© 2023 Veridise Inc.

4.1 Detailed Description of Issues

4.1.1 V-WUSD-VUL-001: Centralization Risk

Severity	Warning	**Commit**	0xbdAd407F77...
Type	Centralization	**Status**	Open
File(s)	`MantaMintableERC20WithBridgeFlag.sol`		
Location(s)	N/A		
Confirmed Fix At			

Similar to many projects, the `MantaMintableERC20WithBridgeFlag` token declares administrator roles that are given special permissions. In particular, it declares an administrator that is given the ability to pause token burns so that they cannot be bridged out of Manta Pacific.

Impact If a private key were stolen, a hacker would have access to sensitive functionality that could compromise the project. For example, a malicious admin could DoS the bridging of tokens out of Manta.

Recommendation As this is a sensitive operation, we would encourage the developers to utilize a decentralized governance or multi-sig contract as opposed to an EOA as an EOA introduces a single point of failure. Note that the contract also declares a bridge role, but this role will be given to a bridge which will have its own governance and therefore already adheres to this recommendation.

DRAFT

5 Fuzz Testing

5.1 Methodology

Our goal was to fuzz test Wrapped Mountain Protocol USD to assess its compliance with the ERC 20 specification. The fuzing focused on functional correctness i.e, whether the implementation deviates from the intended behavior. We used OrCa as our fuzzer and wrote invariants–logical formulas that should hold after every transaction. We then encoded those invariants as assertions in [V].

5.2 Properties Fuzzed

Table 5.1 describes the invariants we fuzz-tested. The second column describes the invariant informally in English, and the third shows the total amount of compute time spent fuzzing this property. The last column notes whether we found a bug when fuzzing the invariant (✗ indicates no bug was found and ✓ means fuzzing this invariant revealed a bug).

The Veridise auditors devoted a total of 8 compute-hours to fuzzing this protocol, identifying a total of 0 bugs.

Table 5.1: Invariants Fuzzed.

Specification	Invariant	Minutes Fuzzed	Bugs Found
V-WUSD-SPEC-001	ERC20.01: transfer should revert if a user atte...	8	0
V-WUSD-SPEC-002	ERC20.02: Funds should be successfully transfer	8	0
V-WUSD-SPEC-003	ERC20.03: static totalSupply	8	0
V-WUSD-SPEC-004	ERC20.04: allowance/balances	8	0
V-WUSD-SPEC-005	ERC20.05: funds transfer	8	0
V-WUSD-SPEC-006	ERC20.06: no extra modfications	8	0
V-WUSD-SPEC-007	ERC20.07: approve makes appropriate state char	8	0
V-WUSD-SPEC-008	ERC20.08: increaseAllowance correctness	8	0
V-WUSD-SPEC-009	ERC20.09: decreaseAllowance correctness	8	0
V-WUSD-SPEC-010	ERC20.13: burn burns correct amount	8	0
V-WUSD-SPEC-011	ERC20.14: burn reduces total supply	8	0
V-WUSD-SPEC-012	ERC20.15: no unintended side effects in burn	8	0
V-WUSD-SPEC-013	ERC20.16: mint increases totalSupply/balance	8	0
V-WUSD-SPEC-014	ERC20.17: mint has no bad side effects	8	0

DRAFT

© 2023 Veridise Inc.

5.3 Detailed Description of Fuzzed Specifications

5.3.1 V-WUSD-SPEC-001: ERC20.01: transfer should revert if a user attempts to send more funds than they have

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `transfer` should revert if a user attempts to send more funds than they have.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t
inv:  reverted(t.transfer(to, amt), amt > t.balanceOf(sender))
```

DRAFT

5.3.2 V-WUSD-SPEC-002: ERC20.02: Funds should be successfully transferred from sender to to as long as sender != to

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language Funds should be successfully transferred from sender to to as long as `sender` ≠ `to`.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t
inv:  finished(t.transfer(to, amt),
  to != sender |=>
    t.balanceOf(sender) = old(t.balanceOf(sender)) - amt &&
    t.balanceOf(to) = old(t.balanceOf(to)) + amt
)
```

DRAFT

 © 2023 Veridise Inc.

5.3.3 V-WUSD-SPEC-003: ERC20.03: transfer should not modify totalSupply, allowances, or balances other than sender and to

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `transfer` should not modify `totalSupply`, allowances, or balances other than `sender` and `to`.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t, address o1, address o2, address o3
inv:  finished(t.transfer(to, amt),
  o1 != sender && o1 != to |=>
    t.totalSupply() = old(t.totalSupply()) &&
    t.balanceOf(o1) = old(t.balanceOf(o1)) &&
    t.allowance(o2, o3) = old(t.allowance(o2, o3))
)
```

DRAFT

5.3.4 V-WUSD-SPEC-004: ERC20.04: transferFrom should enforce allowance and user balance

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `transferFrom` should enforce allowance and user balance.

`transferFrom` should revert when the amount requested is greater than what the spender owns or beyond the recipient's allowance.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t
inv:  reverted(t.transferFrom(from, to, amt),
  amt > t.balanceOf(from) || (from != sender && amt > t.allowance(from, sender))
)
```

DRAFT

 © 2023 Veridise Inc.

5.3.5 V-WUSD-SPEC-005: ERC20.05: As long as from != to, funds should be transferred from from to to and sender should have their allowance credited

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language As long as `from` ≠ `to`, funds should be transferred from `from` to `to` and `sender` should have their allowance credited.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t
inv:  finished(t.transferFrom(from, to, amt),
  from != to |=>
    t.balanceOf(from) = old(t.balanceOf(from)) - amt &&
    t.balanceOf(to) = old(t.balanceOf(to)) + amt &&
    ( (from != sender && old(t.allowance(from, sender)) != MAX_UINT256) ==>
      t.allowance(from, sender) = old(t.allowance(from, sender)) - amt )
)
```

DRAFT

5.3.6 V-WUSD-SPEC-006: ERC20.06: transferFrom should not modify totalSupply, other allowances, or balances

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `transferFrom` should not modify `totalSupply`, other allowances, or balances.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t, address o1, address o2, address o3, address
    o4
inv:  finished(t.transferFrom(from, to, amt),
  o1 != from && o1 != to && o2 != sender && o3 != from |=>
    t.balanceOf(o1) = old(t.balanceOf(o1)) &&
    t.allowance(from, o2) = old(t.allowance(from, o2)) &&
    t.allowance(o3, o4) = old(t.allowance(o3, o4)) &&
    t.totalSupply() = old(t.totalSupply())
)
```

DRAFT

 © 2023 Veridise Inc.

5.3.7 V-WUSD-SPEC-007: ERC20.07: approve makes appropriate state changes

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `approve` makes appropriate state changes.

`approve` should never finish in a state where the allowance of the `spender` is not equal to the given `amount`. `totalSupply`, other allowances and balances should not be modified.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t, address o1, address o2, address o3
inv:  finished(t.approve(spender, amt),
  o2 != sender && o1 != spender |=>
    t.allowance(sender, spender) = amt &&
    t.allowance(sender, o1) = old(t.allowance(sender, o1)) &&
    t.allowance(o2, o3) = old(t.allowance(o2, o3)) &&
    t.balanceOf(o3) = old(t.balanceOf(o3)) &&
    t.totalSupply() = old(t.totalSupply())
)
```

DRAFT

5.3.8 V-WUSD-SPEC-008: ERC20.08: increaseAllowances makes appropriate state changes

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `increaseAllowances` makes appropriate state changes.

The function should increase a user's allowance by the indicated `amount`. `totalSupply`, other allowances, and balances should not be modified.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t, address o1, address o2, address o3
inv:  finished(t.increaseAllowance(spender, amt),
  o2 != sender && o1 != spender |=>
    t.allowance(sender, spender) = old(t.allowance(sender, spender)) + amt &&
    t.allowance(sender, o1) = old(t.allowance(sender, o1)) &&
    t.allowance(o2, o3) = old(t.allowance(o2, o3)) &&
    t.balanceOf(o3) = old(t.balanceOf(o3)) &&
    t.totalSupply() = old(t.totalSupply())
)
```

DRAFT

 © 2023 Veridise Inc.

5.3.9 V-WUSD-SPEC-009: ERC20.09: decreaseAllowance makes appropriate state changes

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `decreaseAllowance` makes appropriate state changes.

The function should decrease a user's allowance by the indicated amount. `totalSupply`, other allowances, and balances should not be modified.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t, address o1, address o2, address o3
inv:  finished(t.decreaseAllowance(spender, amt),
  o2 != sender && o1 != spender |=>
    t.allowance(sender, spender) = old(t.allowance(sender, spender)) - amt &&
    t.allowance(sender, o1) = old(t.allowance(sender, o1)) &&
    t.allowance(o2, o3) = old(t.allowance(o2, o3)) &&
    t.balanceOf(o3) = old(t.balanceOf(o3)) &&
    t.totalSupply() = old(t.totalSupply())
)
```

DRAFT

5.3.10 V-WUSD-SPEC-013: ERC20.13: burn will revert if a user attempts to burn more than they own or more than their allowance

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `burn` will revert if a user attempts to burn more than they own or more than their allowance.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t
inv:  reverted(t.burn(from, amt),
  amt > t.balanceOf(from) || (from != sender && amt > t.allowance(from, sender))
)
```

DRAFT

 © 2023 Veridise Inc.

5.3.11 V-WUSD-SPEC-014: ERC20.14: burn will reduce the total supply and the balance of from by the indicated amount

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `burn` will reduce the total supply and the balance of from by the indicated amount and only the bridge may burn tokens

Formal This is a modified version of a specification from the Veridise [V] specification library. The specification required modifications since only the bridge may burn tokens and it does not reduce the allowance.

```
vars: MantaMintableERC20WithBridgeFlag t
inv:  finished(t.burn(from, amt),
  t.balanceOf(from) = old(t.balanceOf(from)) - amt &&
  t.totalSupply() = old(t.totalSupply()) - amt &&
  sender = t.bridge()
)
```

DRAFT

5.3.12 V-WUSD-SPEC-015: ERC20.15: burn will not modify another user balance or other allowances

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `burn` will not modify another user's balance or other allowances.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t, address o1, address o2, address o3
inv:  finished(t.burn(from, amt),
  o1 != from && o2 != sender |=>
    t.balanceOf(o1) = old(t.balanceOf(o1)) &&
    t.allowance(from, o2) = old(t.allowance(from, o2)) &&
    t.allowance(o1, o3) = old(t.allowance(o1, o3))
)
```

DRAFT

 © 2023 Veridise Inc.

5.3.13 V-WUSD-SPEC-016: ERC20.16: mint will increase totalSupply and user balance by the indicated amount

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `mint` will increase `totalSupply` and a user's balance by the indicated amount.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t
inv:  finished(t.mint(acc, amt),
  t.balanceOf(acc) = old(t.balanceOf(acc)) + amt &&
  t.totalSupply() = old(t.totalSupply()) + amt
)
```

DRAFT

5.3.14 V-WUSD-SPEC-017: ERC20.17: mint will not modify the balance of another user or any allowances

Minutes Fuzzed	60	Bugs Found	0

Specification

Scope ERC20 tokens.

Natural Language `mint` will not modify another user's balance or any allowances.

Formal This specification is part of the Veridise [V] specification library.

```
vars: MantaMintableERC20WithBridgeFlag t, address o1, address o2, address o3
inv:  finished(t.mint(acc, amt),
  o1 != acc |=>
    t.balanceOf(o1) = old(t.balanceOf(o1)) &&
    t.allowance(o2, o3) = old(t.allowance(o2, o3))
)
```

DRAFT

 © 2023 Veridise Inc.

DRAFT

Glossary

ERC 20 The famous Ethereum fungible token standard. See `https://eips.ethereum.org/EIPS/eip-20` to learn more. 9

smart contract A self-executing contract with the terms directly written into code. Hosted on a blockchain, it automatically enforces and executes the terms of an agreement between buyer and seller. Smart contracts are transparent, tamper-proof, and eliminate the need for intermediaries, making transactions more efficient and secure.. 25

Solidity The standard high-level language used to develop smart contracts on the Ethereum blockchain. See `https://docs.soliditylang.org/en/v0.8.19/` to learn more. 1

DRAFT

© 2023 Veridise Inc.