Veridise. Source Code Verification Report

Hardening Blockchain Security with Formal Methods

FOR

Uniswap V3 Manta Pacific Deployment

Veridise Inc.

October 4, 2023

► **Prepared For:**

Aperture Finance
https://www.aperture.finance

► **Prepared By:**

Himanshu
Benjamin Sepanski

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

► **Version History:**

Oct. 4, 2023	V1
Sep. 29, 2023	Initial Draft

Contents

1 Executive Summary

From Sep. 26, 2023 to Sep. 28, 2023, Aperture Finance engaged Veridise to perform source code verification on their Uniswap V3 Manta Pacific Deployment.

Aperture Finance deployed Uniswap V3 to the Manta Pacific Network on September 13, 2023. To do this, they used the *Uniswap deploy-v3 repository. They configured this using †Manta Pacific's deployment of ‡WETH9. In addition, Aperture Finance deployed §`SequenceUtils.sol`. For a full list of addresses and links to verified source code, see section 2.

Veridise conducted the assessment over 6 person-days, with 2 engineers verifying the source code over 3 days. The source code verification consisted of recreating the expected creation code locally and comparing it to transactions on the Manta Pacific Network via queries to one of the networks ¶JSON-RPC Nodes.

Deployment assessment. The Uniswap V3 Manta Pacific Deployment developers provided addresses for all of thoe deployed contracts, a link to the Uniswap V3 deployment script, the WETH9 contract, and a deployment of `SequenceUtils` on Ethereum.

The Veridise team successfully verified the source code of each contract at the provided addresses, and checked the correct constructor arguments were supplied.

Summary of issues detected. The source code verification identified only one potential issue V-APF-VUL-001, in which invalid addresses are supplied to the `SequenceUtils` contract. This was acknowledged by the developer, as only the `MultiCallUtils` features of `SequenceUtils` will be used.

Recommendations. The developers should document that functions inherited by `SequenceUtils` from `RequireUtils` should not be used, as the supplied constructor arguments are invalid.

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.

* https://github.com/Uniswap/deploy-v3/tree/b7aac0f1c5353b36802dc0cf95c426d2ef0c3252
† https://docs.manta.network/docs/manta-pacific/Quickstart#token-list
‡ https://pacific-explorer.manta.network/address/0x0Dc808adcE2099A9F62AA87D9670745AbA741746
§ https://github.com/0xsequence/wallet-contracts/blob/a1f11455340d5b742ba84f9bc9c8346b6ce1a2c9/src/contracts/modules/utils/SequenceUtils.sol
¶ https://docs.manta.network/docs/manta-pacific/JSON-RPC%20Nodes
https://etherscan.io/address/0xd130B43062D875a4B7aF3f8fc036Bc6e9D3E1B3E#code

2 Project Dashboard

Table 2.1: Application Summary.

Name	Type	Platform
Uniswap V3 Manta Pacific Deployment	EVM	Manta Pacific Network

Table 2.2: Engagement Summary.

Dates	Method	Consultants Engaged	Level of Effort
Sep. 26 - Sep. 28, 2023	Manual & Tools	2	6 person-days

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	1	1
Warning-Severity Issues	0	0
Informational-Severity Issues	0	0
TOTAL	1	1

Table 2.4: Category Breakdown.

Name	Number
Nonexistent Contract	0
Incorrect Contract Bytecode	0
Incorrect Constructor Arguments	1

3 Goals and Scope

3.1 Goals

The engagement was scoped to provide source code verification for Aperture Finance's Uniswap V3 Manta Pacific Deployment. To do so, verified answers to the following questions for each of the supplied contract-address pairs:

- Does a smart contract exist at the provided address?
- Does the creation bytecode supplied in the transaction which creates the contract match the bytecode produced by compiling the contract locally?
- Do the constructor arguments passed during deployment match their expected values?

3.2 Methodology

To address the questions above, the Veridise team reviewed the provided addresses and deployment scripts. They then used the Manta Pacific Network *Block Explorer to identify the transactions which created these contracts.

For the Uniswap contracts, the Veridise team verified that the supplied deployment script came from the Uniswap organization. They then used the deployment script locally to produce the expected creation code for the contracts, modifying the deployment script so that it used the expected constructor arguments (i.e. the Manta Pacific Network addresses).

For the `WETH9` and `SequenceUtils` contracts, the Veridise team compiled the provided source code (see Section 3.3). For the `WETH9` contract, they further verified that this code implements the same functionality as other †popular implementations of WETH9. The only difference identified was in compiler version, and use of the keyword `emit` (used due to changing the compiler version from 0.4.18 to ‡0.4.22).

The Veridise team then used the Manta Pacific Network's §JSON-RPC Nodes to download the transactions, verify that they indeed created a contract at the specified address, and checked that the input data matched the expected creation bytecode and constructor arguments.

3.3 Scope

In this section, we provide each contract's address, creation transaction hash, and creation code block in Table 3.1. We also provide links to the source code of each contract below.

* https://pacific-explorer.manta.network
† Such as https://github.com/makerdao/sai/blob/8b7a7359f40231131218b594fa59ac2bcee5f6ef/src/weth9.sol
‡ See https://soliditylang.org/blog/2018/03/08/solidity-0.4.21-release-announcement/
§ https://docs.manta.network/docs/manta-pacific/JSON-RPC%20Nodes

- **`UniswapV3Factory`.** The associated source code corresponds to the contract included in version 1.0.1 of the uniswap/v3-core NPM package. The artifact (supplied by Uniswap) in this package is the same one which Uniswap's deploy-v3 repository deploys. This artifact was used for comparison against the factory deployed on Manta Pacific.
- **Uniswap V3 Periphery Contracts.** These contracts include
  - `NFTDescriptor`
  - `NonfungiblePositionManager`
  - `NonfungiblePositionManager`
  - `TickLens`
  - `UniswapV3InterfaceMulticall`
  - `UniswapV3Staker`
  - `V3Migrator`
  - `QuoterV2`

  The periphery contracts also include `ProxyAdmin` and `TransparentUpgradeableProxy` as dependencies, which are provided by ¶OpenZeppelin. The associated source code corresponds to the contracts included in version 1.1.1 of the uniswap/v3-periphery NPM package. The artifact (supplied by Uniswap) in this package is the same one which Uniswap's deploy-v3 repository deploys. This artifact was used for comparison against the periphery contracts deployed on Manta Pacific.
- **`SequenceUtils`.** This contract comes from the `SequenceUtils.sol` file and its associated dependencies in the 0xsequence wallet-contracts repository at commit `a1f11455340d5b742-ba84f9bc9c8346b6ce1a2c9`. An identical deployment, supplied to the Veridise team by Aperture Finance, can be found on etherscan.
- **`WETH9`.** This contract was not deployed by the developers. Verified source code can be found on the Manta Pacific Network blockchain explorer**. As mentioned in Section 3.2, this source code is identical in functionality to the WETH9 contract found on Github ††.

Table 3.1: Verified Source Code Table.

Contract Name	Attribute	Value
UniswapV3Factory	Address	0x5bd1F6735B80e58aAC88B8A94836854d3068a13a
	Block	41427
	Tx Hash	0xd5eb54b371909116825c2e6c551e2b3ce082b9e4945f5435f1610d0e683503c7
WETH9	Address	0x0Dc808adcE2099A9F62AA87D9670745AbA741746
	Block	18521
	Tx Hash	0xe012900772e1499c2071cfa4139c6a1a64aa40a66aaa0181d75b86c77f226f0a

¶ https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1-solc-0.7-2/contracts/proxy/ProxyAdmin.sol and https://github.com/OpenZeppelin/openzeppelin-contracts/blob/v3.4.1-solc-0.7-2/contracts/proxy/TransparentUpgradeableProxy.sol
https://etherscan.io/address/0xd130B43062D875a4B7aF3f8fc036Bc6e9D3E1B3E
** https://pacific-explorer.manta.network/address/0x0Dc808adcE2099A9F62AA87D9670745AbA741746/contracts#address-tabs
†† https://github.com/makerdao/sai/blob/8b7a7359f40231131218b594fa59ac2bcee5f6ef/src/weth9.sol

Contract Name	Attribute	Value
UniswapInterfaceMulticall	Address	0x6Cb54E76D7c739430A440A4b2dF97FC4a784EAdf
	Block	41429
	Tx Hash	0x67bb6bb93d33288aa1693bf3a22aca88fb2ca42c6c49503cdebe9fb8cf25bf21
ProxyAdmin	Address	0x2ff78195D50fA975F9c08c8E24B55CD00C6fee43
	Block	41430
	Tx Hash	0x6563fa66de4e8d53f15284c9875e492e871fa75a458b52dbdeb6831e03619e4b
TickLens	Address	0x1AAABDFb88B289093C0951636F980Ed974B02440
	Block	41431
	Tx Hash	0x556887c6b555d465d619b5ad43bec15a7287ecbf999f1bbb47678b415a9b7ab9
NFTDescriptor	Address	0x243533F957B12f7D5De6fF0612B06f50F2921847
	Block	41432
	Tx Hash	0x3d66706d8aaf4d4751b80d27de2c611285ac6cefda57bcca0e535cf0c4d5686e
NonfungibleTokenPositionDescriptor	Address	0xF74d8ff1c885CEECcC9edeFE14F30b6D21173183
	Block	41433
	Tx Hash	0x449f23c6e0cf00b32a94f470c4bf8f16fedb4da574f92708c58d570fc4ab1e6d
TransparentUpgradeableProxy	Address	0xa4041ceDBfcc09A29109b394e24935aB3ae12D2f
	Block	41434
	Tx Hash	0x40d865608618e0d18a8fd592b8cf1c44ccc0bac906962e8c94f3ad0d1837f1d8
NonfungiblePositionManager	Address	0xe77e3F98a386a4C8f8c706A2aCfFdf57e70D06c6
	Block	41435
	Tx Hash	0xb51fcdd6cd475f7f5b86c583c5ace3f0b3802e6dee1f79d05964d3adedc87077
V3Migrator	Address	0x3662E0eFaaf5Ba9Dc728a2a9b499F80Df022c73D
	Block	41436
	Tx Hash	0x48400be29b76c1cc939490f80cb3c336ba85257882e07ded5aa834ab59012d79
UniswapV3Staker	Address	0x2969D4a4cD19c7f49e277295726ec1F6B2D8b0e5
	Block	41438
	Tx Hash	0xd2a7881256e1d74660615a2e3cae7f7dfb6b9718b0b88b3a25e663465cb6db48

Contract Name	Attribute	Value
QuoterV2	Address	0x1e139877CbB99f1fa94BB8763aFc6161cC1dc303
	Block	41439
	Tx Hash	0xd355cae12ca6c5a1f86bf50380070f914607f4663b5ad7436eb3f74857f7f543
SwapRouter02	Address	0x3488d5A2D0281f546e43435715C436b46Ec1C678
	Block	41440
	Tx Hash	0xee8f730d48e3c5a38cb8474df4b2a98deb57dd537804830988acb7bb29879fc8
SequenceUtils	Address	0xF0a625CCF0C8eb40261E3487DF18B6F8933fCfE1
	Block	41501
	Tx Hash	0xfd099f7cc44f53650923b406cb15d63d2a4f799000fcb3b3e96fde2691327c2e

3.4 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 3.2 shows how our auditors weigh this information to estimate the severity of a given issue.

Table 3.2: 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 3.3:

Table 3.3: 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 3.4:

Table 3.4: 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

4 Vulnerability Report

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-APF-VUL-001	Invalid constructor arguments passed to Seq...	Low	Acknowledged

4.1 Detailed Description of Issues

4.1.1 V-APF-VUL-001: Invalid constructor arguments passed to SequenceUtils

Severity	Low	Contract	`SequenceUtils`
Type	Incorrect Constructor Arguments		
Status	Acknowledged		

In the manta pacific deployment, the addresses passed to `SequenceUtils`'s constructor (called `factory` and `mainModule`) are the same ones used in the deployment at address `0xd130B43062D8-75a4B7aF3f8fc036Bc6e9D3E1B3E` on the Ethereum mainnet*. In particular, the values are

- Ethereum
  - `0xf9d09d634fb818b05149329c1dccfaea53639d96` (https://etherscan.io/address/0xf9d09d634fb818b05149329c1dccfaea53639d96)
  - `0xd01f11855bccb95f88d7a48492f66410d4637313` (https://etherscan.io/address/0xd01f11855bccb95f88d7a48492f66410d4637313)
- Manta Pacific
  - `0xf9D09D634Fb818b05149329C1dcCFAeA53639d96` (https://pacific-explorer.manta.network/address/0xf9D09D634Fb818b05149329C1dcCFAeA53639d96)
  - `0xd01F11855bCcb95f88D7A48492F66410d4637313` (https://pacific-explorer.manta.network/address/0xd01F11855bCcb95f88D7A48492F66410d4637313)

As of September 29, 2023, there are no contracts deployed at those addresses on the Manta Pacific Network.

Impact `SequenceUtils` inherits from two different contracts, MultiCallUtils and RequireUtils. `RequireUtils` uses the constructor arguments passed to `SequenceUtils`, while `MultiCallUtils` does not. Any use of the functionality inherited from `RequireUtils` may revert, or behave incorrectly (if contracts are ever deployed to the supplied addresses).

Recommendation Either re-deploy or be sure to indicate to end-users not to use the `RequireUtils` functionality of `SequenceUtils`.

Developer Response The addresses passed to the `SequenceUtils` constructor, named `factory` and `mainModule`, are only relevant to 0xsequence's "Sequence Smart Wallet" product. We only need `SequenceUtils` contract's main logic for the ability to wrap an ethers.js provider into a multicall provider and these parameters are not relevant to our use case.

* https://etherscan.io/address/0xd130B43062D875a4B7aF3f8fc036Bc6e9D3E1B3E

Glossary

AMM Automated Market Maker. 13

Manta Pacific Network A fully EVM-compatible L2. For more information, see `https://pacific.manta.network`. 1

Uniswap One of the most famous deployed AMMs. See `https://uniswap.org` to learn more. 1