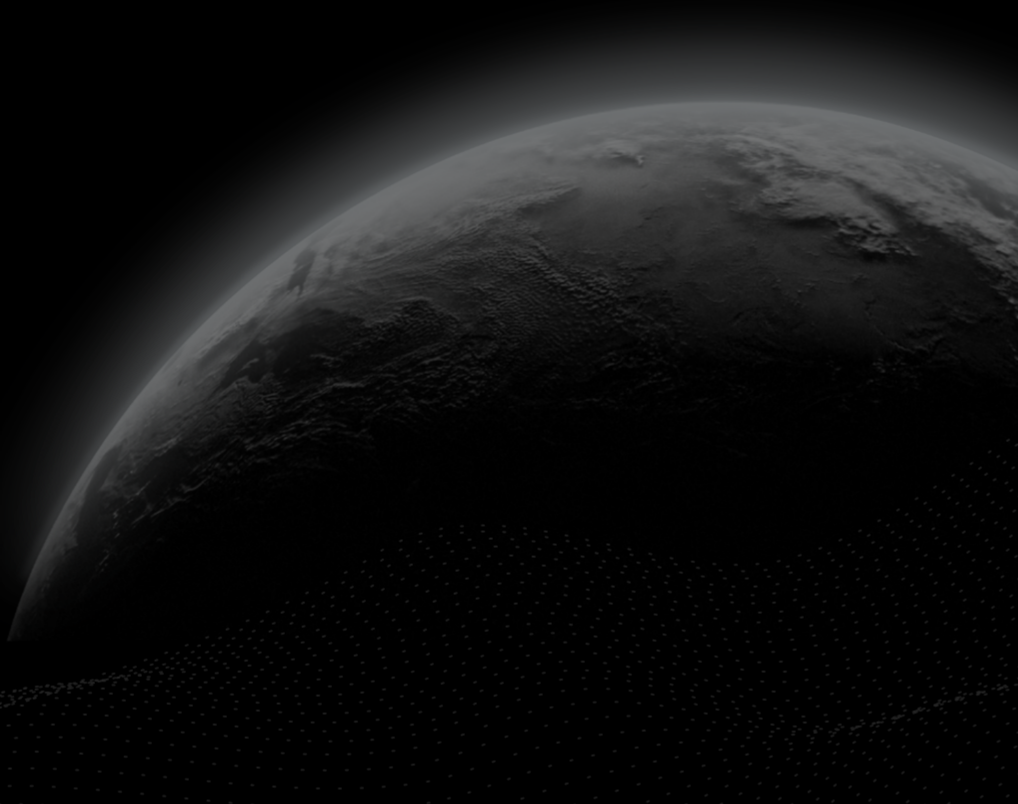
CERTIK

Security Assessment

pyth2wormhole - Ethereum

CertiK Verified on Feb 23rd, 2023



CertiK Verified on Feb 23rd, 2023

pyth2wormhole - Ethereum

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Bridge	Ethereum	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Rust, Solidity	Delivered on 02/23/2023	N/A

CODEBASE

https://github.com/pyth-network/pyth2wormhole

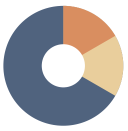
...View All

COMMITS

- b5555b80f74b88bb9f93275ab9ef293e99653f4b
- da1f19bf0b35673773ce642905fcbe3e75611b87
- 8f8eee7c92eb3979d0a9916a9b36acc2d911afb1

...View All

Vulnerability Summary



6	4	0	0	2	0	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined	Unresolved

🟥	0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	1	Major	1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	1	Minor	1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	4	Informational	4 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | PYTH2WORMHOLE - ETHEREUM

CODEBASE | PYTH2WORMHOLE - ETHEREUM

Repository

https://github.com/pyth-network/pyth2wormhole

Commit

- b5555b80f74b88bb9f93275ab9ef293e99653f4b
- da1f19bf0b35673773ce642905fcbe3e75611b87
- 8f8eee7c92eb3979d0a9916a9b36acc2d911afb1
- 01c46619852925d522ab06703d43f1e27442a106

AUDIT SCOPE | PYTH2WORMHOLE - ETHEREUM

48 files audited ● 3 files with Acknowledged findings ● 2 files with Resolved findings ● 43 files without findings

ID	File	SHA256 Checksum
● PYP	pyth/Pyth.sol	678153a18862383bb994e69f742891ab1beb819d2f5fd96fb9b22589564b10d3
● PUB	pyth/PythUpgradable.sol	39a8b91648a6fb760ecb09c66284266885c76ff9f0d442631ff56e5758b84537
● SEU	wormhole/Setup.sol	8602d05c8d48dce15f9788dff708a8c78035553e2b2f134af4b9815bf78a6bb4
● PSB	pyth/PythSetters.sol	8e30fdeba149b6ee07b81360179956fdaad453b2a9dad8fc9e1d55d9e3342c2d
● PSU	pyth/PythState.sol	689267de4c9d23d37bca49575ed70e97bbbafb31270cd827ee34e7e4d17ba226
● BLB	libraries/external/BytesLib.sol	1b6f2ba238f9af311f917ddbf412edc565cfde02398d08727e8bbb98ad14d819
● MPP	pyth/mock/MockPythProxyUpgrade.sol	1182a99c99b247c8adf11cbaab341cdb0acc1df48c6c68a0bb0e17fc27eeacc1
● PGB	pyth/PythGetters.sol	0c551785a124e638d3245463428163a1b0df5901d8b401636d25d89de8c70ef6
● PIS	pyth/PythInternalStructs.sol	f268e9d9639a00f3673432f023479da02710bfe086e2a30772a1065d1fd9c3d0
● RGB	wormhole-receiver/ReceiverGetters.sol	fd0b56b8804ae7f7f72e2030c052cfc4392c36ed5ffecfda438deb68a99829bf
● RGU	wormhole-receiver/ReceiverGovernance.sol	7fd7e2e6981430491f5d14df9ae4a6752f7681585cc82f6ae4c62a9e35cdbc8d
● RGS	wormhole-receiver/ReceiverGovernanceStructs.sol	6ec955f2afc91fd3bcd24fcfa7011e991db3008db1f624ba7e6144ebfcc24522
● RIB	wormhole-receiver/ReceiverImplementation.sol	1b968bbfdab3b1a4f81ce406e6459e923f5d284daa6f597b6a500a46c9e8cac3
● RMB	wormhole-receiver/ReceiverMessages.sol	015f5415506408d5ba537a52afe6d2225fbe23e58008498c8a9ee36af5dafe88

ID	File	SHA256 Checksum
RSB	wormhole-receiver/ReceiverSetters.sol	7bc8ee8c05ba0006e7b32bf975b366f928941902e59410782cb71e1db6a654ad
RSU	wormhole-receiver/ReceiverSetup.sol	f57449d83cb831bb6127a916f3a1da2b7218f9706d5dc7a6d981816ca1761752
RSH	wormhole-receiver/ReceiverState.sol	a6093497b35f95e16d3bc84ca4eccd30f862bf78e1e980a6bcc367747e67eb8d
RST	wormhole-receiver/ReceiverStructs.sol	a9f220e72442d4c376ea49b5e651e2a9ba60260d73ccf38c14bde7f531cdb023
WRB	wormhole-receiver/WormholeReceiver.sol	cf4795dd42b42a82dd0ed3e4caf5efb258d5fc24c2806ba81a61a49f1a8d0d22
IWB	wormhole/interfaces/IWormhole.sol	7307fccee8d2f9fbe51e95d10822d3e386fa60cd1d721561ac58d2ade5df750b
MIB	wormhole/mock/MockImplementation.sol	a02e0eba3fc59e704d88841f489a7a30e70c67e5d363fd40a222bd6da2e640be
GET	wormhole/Getters.sol	91d24680fc1885a1004de52b0f4a28501a2d630713c056cb9b83a1f2e92c44dd
GOE	wormhole/Governance.sol	fec9ef082f1a655060bacb9ee1151dcd698bdeaaeb6880e58a40213f9e822cbc
GSB	wormhole/GovernanceStructs.sol	3fc5b78c1137d192dfbe1fd2b7e3f4470b1d77dda4c22abd201408d2a498c45a
IMP	wormhole/Implementation.sol	cf5bb644f3c5644a3fa34c6e605f8e069e220ebf265782bf7404c25444d933bc
MES	wormhole/Messages.sol	e679decfe2143748af45fd8b3520a7310a08d91a6abddaab51d9d5a3da31750b
SET	wormhole/Setters.sol	5ddca9c7addeea7e4c95459b3125ffc4456ef942dd4929bd0ed82d1fe54335e9
STW	wormhole/State.sol	ab237ec95c2e4dc6ca650ea4f3d8874111fdd53b452406578e14e15313b634bd
STR	wormhole/Structs.sol	d6da02e4ddf08e94417e007863b4b8904084481e83587acfe6b134061ee1a98a
WOH	wormhole/Wormhole.sol	5e57e8d9cf7cf0738e1404e57e18cd3f21e81b703eafaaa18cbad3ed57b7e9f2
MIG	Migrations.sol	d38ffc211dbf5507f18f2afcd8e1c9dd34e790f2c3125fd33965443c2977d639

ID	File	SHA256 Checksum
● MPU	pyth/mock/MockPythProxyUpgrade.sol	1182a99c99b247c8adf11cbaab341cdb0acc1df48c6c68a0bb0e17fc27eeacc1
● PYY	pyth/Pyth.sol	835aceb1741ad56ce15e8bcdb3432813191bd62eabf42442cff41f64d90ce90c
● PGU	pyth/PythGetters.sol	4513098506a849adb6599b1c3192cc05a93bfd2c74b3e64c16dab4cfef854423
● PYI	pyth/PythInternalStructs.sol	c88862c8adf2e9f15c4138ca66d756e8bcb75ea77f41b2182dc3e1cc8431fdf0
● PSH	pyth/PythSetters.sol	6db77b43a42c7e7eb94cd04e0b2562bbf2cd75ebb83332e0c08ffad4d0fc1531
● PST	pyth/PythState.sol	d917de8d7fb226b1cac421957736e0edd7fb03e5ac725228220705b6bb2a49f3
● PUU	pyth/PythUpgradable.sol	809d5fdd3d686a1e09ac68230bf02c16d0f60db15fd9a4499fe879b040b95893
● RGH	wormhole-receiver/ReceiverGetters.sol	fd0b56b8804ae7f7f72e2030c052cfc4392c36ed5ffecfda438deb68a99829bf
● RGT	wormhole-receiver/ReceiverGovernance.sol	7fd7e2e6981430491f5d14df9ae4a6752f7681585cc82f6ae4c62a9e35cdbc8d
● REC	wormhole-receiver/ReceiverGovernanceStructs.sol	6ec955f2afc91fd3bcd24fcfa7011e991db3008db1f624ba7e6144ebfcc24522
● RIU	wormhole-receiver/ReceiverImplementation.sol	1b968bbfdab3b1a4f81ce406e6459e923f5d284daa6f597b6a500a46c9e8cac3
● RMU	wormhole-receiver/ReceiverMessages.sol	015f5415506408d5ba537a52afe6d2225fbe23e58008498c8a9ee36af5dafe88
● RSI	wormhole-receiver/ReceiverSetters.sol	7bc8ee8c05ba0006e7b32bf975b366f928941902e59410782cb71e1db6a654ad
● RSG	wormhole-receiver/ReceiverSetup.sol	f57449d83cb831bb6127a916f3a1da2b7218f9706d5dc7a6d981816ca1761752
● RSK	wormhole-receiver/ReceiverState.sol	a6093497b35f95e16d3bc84ca4eccd30f862bf78e1e980a6bcc367747e67eb8d
● RSR	wormhole-receiver/ReceiverStructs.sol	a9f220e72442d4c376ea49b5e651e2a9ba60260d73ccf38c14bde7f531cdb023
● WRU	wormhole-receiver/WormholeReceiver.sol	cf4795dd42b42a82dd0ed3e4caf5efb258d5fc24c2806ba81a61a49f1a8d0d22

APPROACH & METHODS | PYTH2WORMHOLE - ETHEREUM

This report has been prepared for Wormhole to discover issues and vulnerabilities in the source code of the pyth2wormhole - Ethereum project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

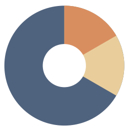
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | PYTH2WORMHOLE - ETHEREUM



6	0	1	0	1	4
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for pyth2wormhole - Ethereum. Through this audit, we have uncovered 6 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
PUB-01	**Centralized Control Of Upgradeable Contracts**	**Centralization / Privilege**	**Major**	**● Acknowledged**
SEU-01	Lack Of Sanity Check	Volatile Code	Minor	● Acknowledged
PSB-01	Lack Of Event Emitting	Coding Style	Informational	● Resolved
PSU-01	Unnecessary `payable` Address Type	Language Specific	Informational	● Resolved
PYP-01	Potential Incorrect Decoding Process	Logical Issue	Informational	● Resolved
PYP-02	Lack Of Authority Checks	Logical Issue	Informational	● Resolved

PUB-01 | CENTRALIZED CONTROL OF UPGRADEABLE CONTRACTS

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	**pyth/PythUpgradable.sol (b5555b80f74b88bb9f93275ab9ef293e99653f4b): 11**	● **Acknowledged**

Description

The contract `PythUpgradable` is an upgradeable contract, the owner can upgrade the contract without the community's commitment. If an attacker compromises the account, he/she can change the implementation contract, leading to unexpected loss.

Exploit scenario:

1. A hacker compromises the private key of the proxy owner account;
2. The hacker updates the implementation contract with malicious functionality;
3. The hacker executes the malicious functionality through the proxy contract.

Update on 11/24/2022:

In the commit da1f19bf0b35673773ce642905fcbe3e75611b87, the protocol introduced two privileged functions that can be invoked by the `_owner` of the contract. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.

- `addDataSource()` adds additional data sources;
- `removeDataSource()` removes the specified data source.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Pyth Team, 11/24/2022]:

The Pyth team acknowledged this issue and stated that the team currently implements a governance mechanism in place to upgrade the contracts.

SEU-01 | LACK OF SANITY CHECK

Category	Severity	Location	Status
Volatile Code	● Minor	wormhole/Setup.sol (b5555b80f74b88bb9f93275ab9ef293e99653f4b): 26, 32, 34	● Acknowledged

Description

In the `Setup.sol` contract, the following initial settings in the `setup()` function are recommended to be verified as non-zero values:

- `address implementation`
- `address[] memory initialGuardians`
- `bytes32 governanceContract`

Recommendation

We advise the client to check that the addresses are not zero by adding corresponding checks to all the above-mentioned parameters in the `setup()` function. Example:

```
require(implementation != address(0), "implementation's address must not be address(0)");
```

Alleviation

[Pyth Team, 11/24/2022]:

The team acknowledged the finding and decided not to change the current codebase.

PSB-01 | LACK OF EVENT EMITTING

Category	Severity	Location	Status
Coding Style	● Informational	pyth/PythSetters.sol (b5555b80f74b88bb9f93275ab9ef293e99653f4b): 9, 13, 17, 21	● Resolved

Description

Functions that affect the status of sensitive variables should emit events as notifications to customers.

Example: In the contract `PythSetters` :

- function `setPyth2WormholeChainId()` sets `pyth2WormholeChainId` ;
- function `setPyth2WormholeEmitter()` sets `pyth2WormholeEmitter` ;
- function `setWormhole()` sets `wormhole` ;
- function `setLatestPriceInfo()` sets `latestPriceInfo` .

Recommendation

Recommend adding events for sensitive actions in the aforementioned functions and emit them in the functions.

Alleviation

[Pyth Team, 11/24/2022]:

The team resolved this issue by emitting events in the commit 01c46619852925d522ab06703d43f1e27442a106

PSU-01 | UNNECESSARY `payable` ADDRESS TYPE

Category	Severity	Location	Status
Language Specific	● Informational	pyth/PythState.sol (b5555b80f74b88bb9f93275ab9ef293e99653f4b): 10	● Resolved

Description

In the `PythStorage` contract, the address `wormhole` has a `payable` attribute. However, the current contracts do not send any ETH to the `wormhole` address.

```
    struct State {
        address payable wormhole;
        //...
    }
```

Recommendation

We advise the client to change the variable from type `address payable` to `address` to increase the legibility of the code.

```
    struct State {
        address wormhole;
        //...
    }
```

Alleviation

[Pyth Team, 11/24/2022]:

The team heeded the advice and resolved this issue in commit b062cd51fa4f1a256136c8f95b7c8daac5bcf525.

PYP-01 | POTENTIAL INCORRECT DECODING PROCESS

Category	Severity	Location	Status
Logical Issue	● Informational	pyth/Pyth.sol (b5555b80f74b88bb9f93275ab9ef293e99653f4b): 120~123	● Resolved

Description

In the `parseBatchPriceAttestation()` function, the passed-in parameter `encoded` will be decoded into a struct `BatchPriceAttestation` based on a fixed pattern. The `index` variable serves as the pointer and will move forward after a value is decoded. For example,

- to decode the `magic` value, which is a `uint32` type, it reads 32 bits from the `index` and moves forward the `index` by 4;
- to decode `versionMajor` value, which is a `uint16` type, it reads 16 bits from the `index` and moves forward the `index` by 2.

However, when decoding the `payloadId`, which is a type of `uint8`, it reads 8 bits from the index, but moving forward the `index` by `bpa.header.hdrSize` instead of `1`.

```
        bpa.header.payloadId = encoded.toUint8(index);

        // Skip remaining unknown header bytes
        index += bpa.header.hdrSize;
```

Recommendation

We recommend moving forward the `index` accurately when decoding.

Alleviation

[Pyth Team, 11/24/2022]:

The team confirm that it is an intended design. The `hdr_size` marks the count of remaining header bytes. The `payload_id` is the only header field that comes after. The number of steps for payload_id is accounted for in the `hdr_size`.

PYP-02 | LACK OF AUTHORITY CHECKS

Category	Severity	Location	Status
Logical Issue	● Informational	pyth/Pyth.sol (b5555b80f74b88bb9f93275ab9ef293e99653f4b): 17	● Resolved

Description

In the contract `Pyth.sol` , the function `initialize()` could be invoked by anyone to update the key state variables `wormhole` , `pyth2WormholeChainId` , and `pyth2WormholeEmitter` .

For example, the `wormhole` address is used to parse and verify the data `encodedVm` in the function `updatePriceBatchFromVm()` . Arbitrary calls to the `initialize()` function and updating the key state variables will result in an unexpected result.

Recommendation

We assume that the `Pyth` contract will only be used as a parent contract and will never be used alone. In that case, we recommend marking it as an `abstract contract` to ensure it cannot be deployed directly.

Alleviation

[Pyth Team, 11/24/2022]:

The team heeded the advice and resolved this issue in commit 8f8eee7c92eb3979d0a9916a9b36acc2d911afb1.

OPTIMIZATIONS | PYTH2WORMHOLE - ETHEREUM

ID	Title	Category	Severity	Status
PYP-03	Tautology	Gas Optimization	Optimization	● Acknowledged

PYP-03 | TAUTOLOGY

Category	Severity	Location	Status
Gas Optimization	● Optimization	pyth/Pyth.sol (b5555b80f74b88bb9f93275ab9ef293e99653f4b): 101	● Acknowledged

Description

The linked statements compare a `uint16` variable to be greater than or equal to 0. These statements will always return `true` because unsigned integers cannot be less than 0.

Recommendation

Recommend fixing the redundant comparison by removing the unnecessary check.

Alleviation

[Pyth Team, 11/24/2022]:

The team acknowledged the finding and decided not to change the current codebase.

APPENDIX | PYTH2WORMHOLE - ETHEREUM

Finding Categories

Categories	Description
Centralization / Privilege	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as functions restricted to a privileged set of users.
Gas Optimization	"Gas" is used here as generic term in DLT world, that can differ from chain to chain. Finding indicates that computational, storage resources can be saved, for benefit of users and efficiency of chain. Also in some cases, being not resourceful may lead to DoS attacks.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as unintended deviations from the original business logic of the code base.
Volatile Code	Specifics may differ between runtime environment and (virtual) machine, however in principle findings indicate that assumptions that one may assume by reading code, may not hold, as there maybe other factors that may influence the state, which may lead to other issues (e.g. logical or control flow issues).
Language Specific	Language Specific findings are issues that would only arise within Rust, e.g., Needless borrow.
Coding Style	Coding Style findings suggest how to increase the readability and, thus, the codebase's maintainability. Usually, they do not affect the generated byte code.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.