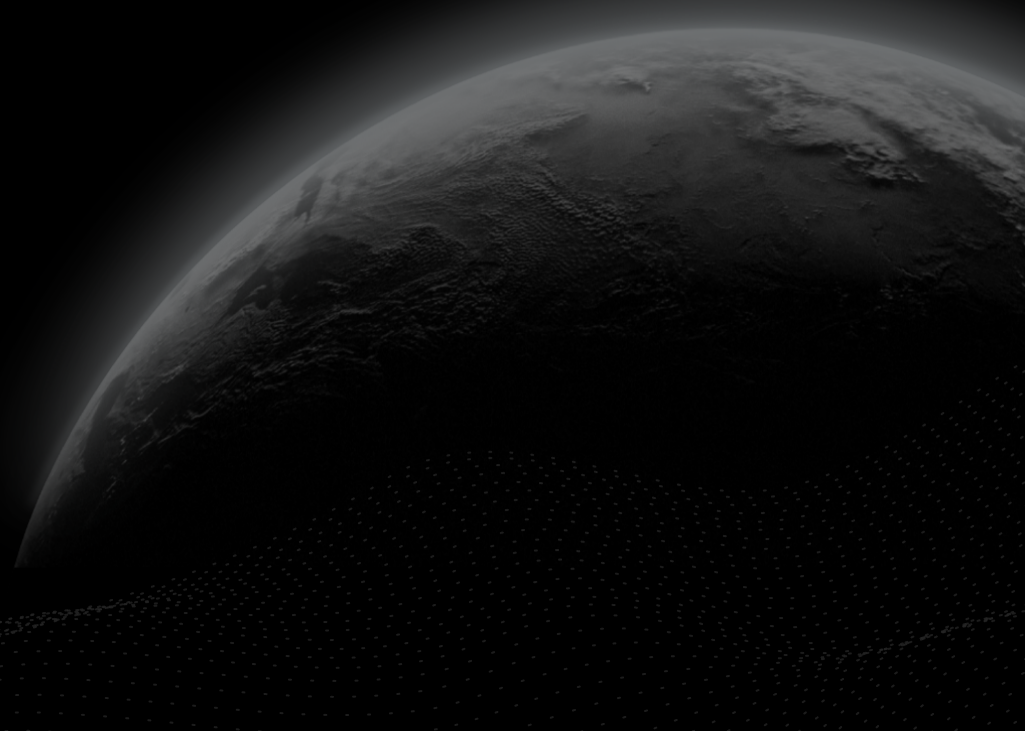
CERTIK

Security Assessment

Pyth-client

CertiK Verified on Nov 30th, 2022



CertiK Verified on Nov 30th, 2022

Pyth-client

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
Oracle	Wormhole	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
C++	Delivered on 11/30/2022	N/A

CODEBASE
https://github.com/pyth-network/pyth-client
...View All

COMMITS
c63246542125cc3a75218cb1debf9755457563d6
...View All

Vulnerability Summary

11	4	0	0	7	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.
2	Major	2 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.
1	Medium	1 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
5	Minor	2 Resolved, 3 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.
3	Informational	1 Resolved, 2 Acknowledged	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 | PYTH-CLIENT

CODEBASE | PYTH-CLIENT

Repository

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

Commit

c63246542125cc3a75218cb1debf9755457563d6

AUDIT SCOPE | PYTH-CLIENT

14 files audited ● 3 files with Acknowledged findings ● 2 files with Resolved findings ● 9 files without findings

ID	File	SHA256 Checksum
ORA	program/src/oracle/oracle.c	02a4fb9061bcd927dbe4b67510b3faf4925ed82faad0beb0e4995127835ea765
ORC	program/src/oracle/oracle.h	f4ae50bb82279c415a0d3e8e922c1723336f0c1693d426614e505adb223747c3
PDO	program/src/oracle/pd.h	062d7ec298a0a16f2610db15acfa333060662d7a7c11bb59edae3753e76e20a3
PRC	program/src/oracle/model/price_model.h	c803f1d625623b34982f5b6457fdbdf450350f9962cc879fef2edb9bbdb644e3
PRN	program/src/oracle/util/prng.h	9cc93381ec43bd87a3edd5ca636abf2b1ef80e377558a003532ea7013212eecc
PRI	program/src/oracle/model/price_model.c	f6d7444a1c1a67ffe6f83b1a63cd99fc27c07d5e206dbd7c8be4b92c015aba76
SOR	program/src/oracle/sort/sort_stable_base_gen.c	97df993dc7a374200bbb077ba730cfb88e0632cd41d2c79ed4f75a02df819cef
ALI	program/src/oracle/util/align.h	a9cf8c3e189762ce5ef5835c17be7ddb2dc811ee2f9644cad04c0e12e45f0bbd
AVG	program/src/oracle/util/avg.h	b93e0d4bf199109e8076ff81d2dec8ca6d7345908b7f5f79854b4cb56d2afec5
COM	program/src/oracle/util/compat_stdint.h	1b5134a2e9a960d054b3268f2a246747bb55286fdc16dd4bee51ec6386c7a456
HAS	program/src/oracle/util/hash.h	be81d745243fe7b31b23b8b5dd7256e849a96beb5c4c1092e5c48a06d308fac4
SAR	program/src/oracle/util/sar.h	9eb74064fd87ecdcfb9388aa71283af407677381d548eaedddc1f498a771b7c3
UTI	program/src/oracle/util/util.h	55a8e4a0a4bb8a69928521f8775f37da51f1e729184e55d5fc12bdd3bb645e8a
UPD	program/src/oracle/upd_aggregate.h	f95703d18269b376edf16e66e15afcf8d975cdbeb2d614ce1fb82e73a17da5af

APPROACH & METHODS | PYTH-CLIENT

This report has been prepared for Pyth Network to discover issues and vulnerabilities in the source code of the Pyth-client 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.

REVIEW NOTES | PYTH-CLIENT

- The audit used this commit.
- A preliminary report was provided on Aug 2nd, 2022.
- The fixes in the final report were audited in this commit, but no re-audit of the additional changes made was performed.

FINDINGS | PYTH-CLIENT

11	0	2	1	5	3
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Pyth-client. Through this audit, we have uncovered 11 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
ORA-01	**Upgradable Program Centralization Risk**	**Centralization / Privilege**	**Major**	**Acknowledged**
ORA-02	**Centralization Related Risks**	**Centralization / Privilege**	**Major**	**Acknowledged**
ORA-03	Lack Of Rent Validation	Language Specific	Medium	Resolved
ORA-04	Weak Argument Check In `set_min_pub()`	Volatile Code	Minor	Resolved
ORA-05	`num_` Validation In The Wrong Place	Volatile Code	Minor	Acknowledged
ORA-06	Incorrect Checks On `data_len`	Volatile Code	Minor	Resolved
ORA-07	Missing Input Validation	Inconsistency	Minor	Acknowledged
PDO-01	Lack Of Zero Validation For `n2->v_`	Inconsistency	Minor	Acknowledged
ORA-08	Redundant Backslash	Coding Style	Informational	Acknowledged
ORC-01	Typos	Inconsistency	Informational	Resolved

ID	Title	Category	Severity	Status
ORC-02	Missing Error Messages	Coding Style	Informational	Acknowledged

ORA-01 | UPGRADABLE PROGRAM CENTRALIZATION RISK

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**program/src/oracle/oracle.c (base): 1**	**Acknowledged**

Description

A Solana program can be deployed on the mainnet as:

- final: the code cannot be updated.
- upgradable: `BPFLoaderUpgradeable` needs to be set as the program owner and an `upgrade authority`, which is a user account, is given.

In case the `Pyth` program is deployed as upgradable, the `upgrade authority` has the privilege to update the implementation of the program at his/her will.

Any compromise to the `upgrade authority` account may allow a hacker to take advantage of this authority and control the implementation of the program and therefore execute potential malicious functionalities in the program.

Recommendation

Our recommendation depends on the team's intentions that we invite to clarify.

If the `Pyth` program is going to be deployed as final, no further actions are needed to address the finding.

Otherwise, we recommend that the team make efforts to restrict access to the private key of the `upgrade authority` account. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

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 of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key being compromised;
  AND

- A medium/blog link for sharing the timelock 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 and multi-signers addresses, and DAO information with the public audience.

Permanent:

Deploying the program as `final` can fully resolve the risk.

Note: we recommend the project team consider 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.

For remediation and mitigated status, please provide the following information:

- Provide the account address with ALL the multi-signer addresses for the verification process.
- Provide a link to the medium/blog with all of the above information included

Alleviation

`[CertiK]`: The team acknowledged the finding and start to use a multi-sig approach.

ORA-02 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	**Major**	**program/src/oracle/oracle.c (base): 39, 68~69, 109~110, 157, 209~210, 256, 301, 332, 380, 421, 446**	**Acknowledged**

Description

The owner of the private key from the `mapping` account has the ability to add the next account in the chain, as well as adding accounts of products, prices, and as a consequence, add and delete publishers. This allows the owner of the private key to fully control the data provided by oracle for its customers, which creates a risk of centralization in case of compromise.

Recommendation

We recommend that the team make efforts to restrict access to the private key of the `mapping` account. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

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 of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key being compromised;
  AND
- A medium/blog link for sharing the timelock 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 and multi-signers addresses, and DAO information with the public audience.

Note: we recommend the project team consider the long-term solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

For remediation and mitigated status, please provide the following information:

- Provide the account address with ALL the multi-signer addresses for the verification process.
- Provide a link to the medium/blog with all of the above information included

We suggest improving and securing the admin logic in a decentralized way.

Additionally, we invite the team to document it with both text and unit tests, and scenario integration testing to explicitly describe why the design decision was taken.

Alleviation

`[CertiK]` : The team acknowledged the finding and will start to use a multi-sig approach in the future.

ORA-03 | LACK OF RENT VALIDATION

Category	Severity	Location	Status
Language Specific	Medium	program/src/oracle/oracle.c (base): 14, 24	Resolved

Description

In connection with the rent for storing data in the network, the new created accounts must have a certain number of lamports on their balance, depending on the amount of data.

Recommendation

We recommend adding a validation for the minimum number of lamports on new accounts sufficient for rent-exemption.

Alleviation

Resolved in the latest main branch.

ORA-04 | WEAK ARGUMENT CHECK IN `set_min_pub()`

Category	Severity	Location	Status
Volatile Code	Minor	program/src/oracle/oracle.c (base): 292~293, 322, 370, 482	Resolved

Description

The only input validation is a check that sees if the size of the input variables satisfies `prm->data_len`. This check is not rigorous enough and as long as the data size matches, any piece of data can be inputted in the `set_min_pub()` function.

Recommendation

We recommend implementing more thorough checks such as having input validation for the `ver_`, `cmd_` and `min_pub_` in the `cmd_set_min_pub_t` check. And for the `cmd_upd_price_t` check, have input validation for the `ver_`, `cmd_`, `status_`, `unused_`, `price_`, `conf_` and `pub_slot_`.

Alleviation

Remediated in the latest main branch.

ORA-05 | `num_` VALIDATION IN THE WRONG PLACE

Category	Severity	Location	Status
Volatile Code	Minor	program/src/oracle/oracle.c (base): 352~354	Acknowledged

Description

The check `sptr->num_ >= PC_COMP_SIZE` is after using `num_` in loop which can cause `comp_[i]` overflow. Since the incrementation of `num_` occurs after checking we have assigned minor severity to this finding.

Recommendation

We recommend to check `sptr->num_ >= PC_COMP_SIZE` above in the account validation code section like this:

```
339  if ( sptr->magic_ != PC_MAGIC ||
340       sptr->ver_ != cptr->ver_ ||
341       sptr->type_ != PC_ACCTYPE_PRICE ||
342       sptr->num_ >= PC_COMP_SIZE ) {
343     return ERROR_INVALID_ARGUMENT;
344   }
```

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

ORA-06 | INCORRECT CHECKS ON `data_len`

Category	Severity	Location	Status
Volatile Code	Minor	program/src/oracle/oracle.c (base): 21~22, 30	Resolved

Description

In the `oracle.h` file, there are multiple assertions on struct sizes, for example:

```
static_assert( sizeof( pc_map_table_t ) == 20536, "" );
```

As a result `ka->data_len >= dlen` in the functions `valid_signable_account` and `valid_writable_account` should instead be a strict check on size:

```
ka->data_len == dlen;
```

Recommendation

We recommend changing

```
ka->data_len >= dlen
```

to

```
ka->data_len == dlen
```

Alleviation

Remediated in the latest main branch.

ORA-07 | MISSING INPUT VALIDATION

Category	Severity	Location	Status
Inconsistency	Minor	program/src/oracle/oracle.c (base): 551~552	Acknowledged

Description

In the `dispatch()` function, there is a check to see if the `data_len` is less than `sizeof(cmd_hdr_t)`, however there isn't a check to see if `data_len` is larger.

Recommendation

We recommend adding this to the input validation:

```
551   if (prm->data_len != sizeof(cmd_hdr_t) ) {
552      return ERROR_INVALID_ARGUMENT;
553    }
```

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

PDO-01 | LACK OF ZERO VALIDATION FOR `n2->v_`

Category	Severity	Location	Status
Inconsistency	Minor	program/src/oracle/pd.h (base): 88~89	Acknowledged

Description

In the function `pd_div` there is not a check for `n2->v_ == 0`.

Recommendation

We recommending adding a check that will throw for `n2->v_ == 0` to avoid errors when dividing by `v2`.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

ORA-08 | REDUNDANT BACKSLASH

Category	Severity	Location	Status
Coding Style	Informational	program/src/oracle/oracle.c (base): 148~149	Acknowledged

Description

The final continuation backslash is redundant as it is on the last line of the statement:

```
142 #define PC_ADD_STR \
143   tag = (pc_str_t*)src;\
144   tag_len = 1 + tag->len_;\
145   if ( &src[tag_len] > end ) return ERROR_INVALID_ARGUMENT;\
146   sol_memcpy( tgt, tag, tag_len );\
147   tgt += tag_len;\
148   src += tag_len;\
```

Recommendation

We recommend removing the continuation backslash at the end of line 148:

```
142 #define PC_ADD_STR \
143   tag = (pc_str_t*)src;\
144   tag_len = 1 + tag->len_;\
145   if ( &src[tag_len] > end ) return ERROR_INVALID_ARGUMENT;\
146   sol_memcpy( tgt, tag, tag_len );\
147   tgt += tag_len;\
148   src += tag_len;
```

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged.

ORC-01 | TYPOS

Category	Severity	Location	Status
Inconsistency	Informational	program/src/oracle/model/price_model.h (base): 48, 50, 71, 76; program/src/oracle/oracle.c (base): 170; program/src/oracle/sort/tmpl/sort_stable.c (base): 164~165; program/src/oracle/util/prng.h (base): 18, 68, 310	Resolved

Description

There are multiple typos in the code base, the corrections are as follows:

1. In the file: `sort_stable.c` : "interaction" should be "iteration".
2. In the file: `price_model.h` : "srcatch" should be "scratch". "reducd" should be "reduce". [0,cnt) should be [0,cnt].
3. In the file: `avg.h` , "implict" should be "implicit".
4. In the file: `prng.h` , "faciliate" should be "facilitate". "alingment" should be "alignment". "occassionally" should be "occasionally".
5. In the file: `oracle.c` , "ssign" should be "sign".

Recommendation

We recommend fixing the typos.

Alleviation

Resolved in the latest main branch.

ORC-02 | MISSING ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	Informational	program/src/oracle/oracle.h (base): 62~63, 73, 88, 97, 111, 123, 133, 143, 175, 264, 272, 281, 291, 301, 310, 319, 328, 341, 354, 366	Acknowledged

Description

The **assert** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We recommend adding error messages that will be passed back to the caller, for example, instead of having this:

```
static_assert( sizeof( pc_pub_key_t ) == 32, "" );
```

Have something like this instead:

```
static_assert( sizeof( pc_pub_key_t ) == 32, "Incorrect Size For pc_pub_key_t" );
```

Alleviation

`[CertiK]` : The team acknowledged the finding and fix it in future versions.

OPTIMIZATIONS | PYTH-CLIENT

ID	Title	Category	Severity	Status
ORA-09	Using `hdr->ver_` Instead Of `PC_VERSION`	Coding Style	Optimization	Resolved
ORA-10	`i` Variable Already Initialized To 0 Before The For Loop	Coding Style	Optimization	Resolved
ORA-11	Test Functions In The Main Code	Optimization	Optimization	Resolved
ORC-03	`size_` Is Not Used In The Program Logic	Optimization	Optimization	Acknowledged

ORA-09 | USING `hdr->ver_` INSTEAD OF `PC_VERSION`

Category	Severity	Location	Status
Coding Style	Optimization	program/src/oracle/oracle.c (base): 51~52, 54, 75, 79, 90, 117, 121, 131, 162, 165, 418, 434	Resolved

Description

The version of `hdr->ver_` is checked in the `dispatch()` function and cannot have a value other than `PC_VERSION`. In this case, it makes sense to use `PC_VERSION` in further logic.

Recommendation

We recommend replacing `hdr->ver_` with the use of `PC_VERSION`.

Alleviation

Remediated in the latest main branch.

ORA-10 `i` VARIABLE ALREADY INITIALIZED TO 0 BEFORE THE FOR LOOP

Category	Severity	Location	Status
Coding Style	Optimization	program/src/oracle/oracle.c (base): 508~509, 510	Resolved

Description

In the checks to verify that the publisher is valid, variable `i` is initialized as 0:

```
508  uint32_t i = 0;
```

However at the start of the `for` loop, `i` is set to 0 again which isn't necessary:

```
510  for( i=0; i != pptr->num_; ++i ) {
```

Recommendation

We recommend the team to refactor the for loop on line 510 to:

```
510  for( ; i != pptr->num_; ++i ) {
```

Alleviation

Remediated in the latest main branch.

ORA-11 | TEST FUNCTIONS IN THE MAIN CODE

Category	Severity	Location	Status
Optimization	Optimization	program/src/oracle/oracle.c (base): 413~476	Resolved

Description

The `init_test()` and `upd_test()` functions are used for testing and are in the main application code.

Recommendation

We recommend removing the test functions from the main application code and use a separate test code for testing in the test environment.

Alleviation

Resolved in the latest main branch.

ORC-03 | `size_` IS NOT USED IN THE PROGRAM LOGIC

Category	Severity	Location	Status
Optimization	Optimization	program/src/oracle/oracle.h (base): 81	Acknowledged

Description

The `size_` in the `pc_map_table` structure is calculated but not used in the program logic.

Recommendation

We recommend to remove `size_` from the `pc_map_table` structure and not to calculate it.

Alleviation

`[CertiK]`: The team acknowledged the finding and decided to remain unchanged due to `size_` might be used in later versions.

APPENDIX | PYTH-CLIENT

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 explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
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 but rather 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.

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.