Security Assessment

APE MARKET

Oct 21st, 2021

Table of Contents

Summary

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

- 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.

Overview

Project Summary

Project Name	APE MARKET
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/ape-market/ape-market-core
Commit	fed836a8c63465d0cfc5a53acc5e1f9862e3b0e1

Audit Summary

Delivery Date	Oct 21, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

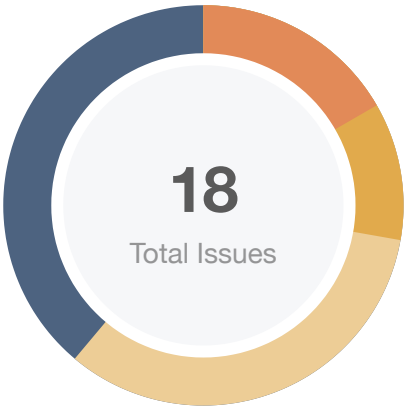
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	3	0	0	0	2	1
Medium	2	0	0	0	0	2
Minor	6	0	0	0	0	6
Informational	7	0	0	0	0	7
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
SAN	nft/SANFT.sol	557fba08aab0007bdff2fc132242a5ee493c4b864e08071714c76095a5d0265e
SAF	nft/SANFTManager.sol	32a8f1cc337f2d9bf1473f55840e5b7da3aaccddd4d5bf6e325095219d66d1b8
ARA	registry/ApeRegistry.sol	057700f542633849a57565aa900a59b25903a14e2d173a88891d407c8cf473d8
FRU	registry/FakeRegistryUser.sol	bfe5a28eb0a8e87e8fb25f42e8954a3bb9b66145b675da686e9c15618e82dd04
RUA	registry/RegistryUser.sol	255475fbd684da14cd4b33e0a728aa7204d98d05bcc8dbd27d544bac7da4586e
SAM	sale/Sale.sol	0730a2813a736abf91ec8e7ecfb663db86db7924bdff705b54cb5ada7b96a22e
SDB	sale/SaleDB.sol	741ef36dd9193fea76099a7959b0d121fbf29567628ae0027c0df6b812e23ea4
SDA	sale/SaleData.sol	4b0cbe95c53684756f7295ac6c9f0406f05d7be2e6c4ceb266f4263345b3133b
SFA	sale/SaleFactory.sol	8f331573169a5c7d39f093a7d188c1744b64631e03ae2dc469ea9463a308464e
SSH	sale/SaleSetupHasher.sol	6dc53eb0cbfb306cde08452711ec13fba4267e4b244d4e79708af6c1ad527332
TRA	sale/TokenRegistry.sol	d8b026a91caa221d2bb60ccd74d34ac3bcaa33c0891d3854b11b25f281549467
PAM	user/Profile.sol	d862dd4e4484b955ba827827c40084e0142dc3efcc47985b185ea571b490dcaf

Findings



Critical	**0** (0.00%)
Major	**3** (16.67%)
Medium	**2** (11.11%)
Minor	**6** (33.33%)
Informational	**7** (38.89%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
APE-01	Financial Models	Logical Issue	Medium	Resolved
ARA-01	Variable never access	Gas Optimization	Informational	Resolved
ARA-02	Centralization Risks	**Centralization / Privilege**	**Major**	Partially Resolved
ARA-03	Optimize loop array	Volatile Code	Minor	Resolved
PAM-01	Potential duplicate identifier	Coding Style	Minor	Resolved
RUA-01	Lack of Zero Address Validation	Volatile Code	Informational	Resolved
RUA-02	Redundant code	Logical Issue	Informational	Resolved
SAF-01	Incorrect judgment condition	Logical Issue	Minor	Resolved
SAF-02	The remaining amount of the `futureToken` did not modified	Logical Issue	Minor	Resolved
SAF-03	No upper limit for fee range	Logical Issue	Medium	Resolved
SAM-01	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Logical Issue	Minor	Resolved
SDA-01	Possible Incorrect Calculation Formula	Mathematical Operations	Minor	Resolved
SDA-02	Redundant ternary expression	Gas Optimization	Informational	Resolved

ID	Title	Category	Severity	Status
SDA-03	Centralization Risk	**Centralization / Privilege**	**Major**	Partially Resolved
SDB-01	State variable never used	Logical Issue	Informational	Resolved
SFA-01	Lack of Input Validation	Logical Issue	Major	Resolved
SSH-01	Potential redundant contract	Logical Issue	Informational	Resolved
TRA-01	Lack of Zero Address Validation	Volatile Code	Informational	Resolved

APE-01 | Financial Models

Category	Severity	Location	Status
Logical Issue	Medium	Global	Resolved

Description

The main functions of `ApeMarket` protocol are as follow:

1. Seller can create a sale by providing the `setUp` of the sale in the contract `SaleFactory`. The seller can transfer the `sellingToken` to the `sale` contract and increase the remaining amount of the sale.
2. Investors can pay some tokens whose kind is specified by the seller to get `SANFT` tokens.
3. The Ape Market provides some APIs to allow investors to manage their `SANFT`. The `SANFT` can be merged, split, sold and etc.
4. After the seller call the function `triggerTokenListing()`, the investor can withdraw their `sellingToken` back.

And then, there are some questions:

1. In the function `SANFTManager.swap()`, the remaining amount of the sale whose id is `tokenSaleId` has already been handled. However, the remaining amount of the sale whose id is `futureTokenSaleId` was not handled. Is that what you want? Please check it.
2. In the function `Sale.vest()`, the amount investor wants to withdraw was compared with the `vestedAmount`. If the amount is more than the `vestedAmount`, investors can not withdraw the `sellingToken`. It is difficult to understand why need the comparison.
3. In the function `SaleData.vestedAmount()`, the function `vestedPercentage()` is used to get the percent of `vested`. The mathematical calculation model of the vested percent is so complex to understand. It may have some problems in the following code:

```
if (step != 0) {
  uint256 ts = (step / 100);
  uint256 percentage = (step % 100) + 1;
  if ((ts * 24 * 3600) + tokenListTimestamp <= currentTimestamp) {
    return uint8(percentage);
  }
}
```

These codes mean that if the time is ok and the step is not 0, the rest of the `extraVestingSteps` will never be accessed. It may make the result of the function `SaleLib.calculateVestedPercentage()` never

changed. Please check this mechanism.

Recommendation

Financial models of blockchain protocols need to be resilient to attacks. They need to pass simulations and verifications to guarantee the security of the overall protocol.

The financial model of this protocol is not in the scope of this audit.

Alleviation

[APE]: A1: Short answer: yes, it is what we want. Long answer: If a sale has a futureTokenSaleId, it is not a regular sale, just a swap following a previous futureToken sale. Any user with tokens from the previous sale, identified by futureTokenSaleId, can swap the future token they purchased into the actual token in this sale. The remaining amount of futureTokenSaleId is managed when futureTokenSaleId was going on. A2: The tokens are locked in the smart agreements and are released as time passes, this is called the vesting process. vestAmount is the amount of tokens that has been released/vested at the time of Sale.vest() is being called. A3: We added comments directly in SaleLib in https://github.com/ape-market/ape-market-core/pull/91

ARA-01 | Variable never access

Category	Severity	Location	Status
Gas Optimization	Informational	registry/ApeRegistry.sol: 33	Resolved

Description

The local variable `changesDone` is never accessed.

Recommendation

Consider deleting it if it is useless.

Alleviation

[APE]: A check of that variable was missed, fixed at https://github.com/ape-market/ape-market-core/pull/89/files#diff-aa059efce55099775 19cb79b5c33016a2b8f119e1a88e7c8ac294667f68cf939R45

ARA-02 | Centralization Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	registry/ApeRegistry.sol: 56, 46, 19	Partially Resolved

Description

In the contract `ApeRegistry`, the role `owner` has the authority over the following function:

- register(): update the registration of the contracts.
- updateContracts()/updateAllContracts(): update the dependencies between all the contracts. Inject the contract registered in this contract to other contracts which depended on the contract. It means that the role `owner` has the authority to use `register` and `updateContracts()` to replace contracts in the project to achieve some goals.

Any compromise to the `owner` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[APE]: We are aware of that. Will replace the owner with a contract controlled by multi-sig before the final report. And we will introduce the DAO mechanism right after we conduct the public sale. This will ensure that there is governance participation from retail investors and not only early presale investors.

Please refer to the following web for details.

- https://ape-market.gitbook.io/untitled/IyW8vBj5Vv6T5k5I4LVZ/

The APE team added a multi-signature owner in the pull request(https://github.com/ape-market/ape-market-core/pull/96).

ARA-03 | Optimize loop array

Category	Severity	Location	Status
Volatile Code	Minor	registry/ApeRegistry.sol: 30	Resolved

Description

```
28 for (uint256 j = 0; j < _contractsList.length; j++) {
29     if (_contractsList[j] == contractHash) {
30         delete _contractsList[j];
31     }
32 }
```

The statement code is used to remove the item which is `_contractsList[j] == contract hash. But there will be a` zero` item left in the array, this array would be longer and longer. And the array may be too long to waste gas.

Recommendation

Consider a different way to remove the element in the array:

1. copy the last item to the target index
2. `pop` this array
3. `break` this `for` loop

Alleviation

[APE]: fixed at https://github.com/ape-market/ape-market-core/pull/89/files#diff-aa059efce55099775 19cb79b5c33016a2b8f119e1a88e7c8ac294667f68cf939R31

PAM-01 | Potential duplicate identifier

Category	Severity	Location	Status
Coding Style	Minor	user/Profile.sol: 21~23	Resolved

Description

The method `_getPseudoAddress` is used to generate a unique address identifier, but is there a possibility of duplicate IDs in such a calculation method?

Recommendation

We suggested using `mapping(address => mapping(address, bool))` to replace the `pseudoAddress`.

Alleviation

[APE]: This is intentional and accurate. It was determined that there is no chance of collision, and is justifiable by the amount of storage saved. As an alternative, we tried `keccak256(abi.encodePacked(addr1, addr2))`. And a mapping (bytes32 => bool), which uses the same amount of storage, but (1) consumes more gas and (2) requires more computation. In fact, the `pseudoAddress` is a sum of two addresses and the order of the addresses is irrelevant. Using the hash, instead, we had to check the two cases (addr1, addr2) and (addr2, addr1).

The recommendation is more clear but takes more storage, and reducing gas cost for the user was one of our primary goals.

RUA-01 | Lack of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Informational	registry/RegistryUser.sol: 24	Resolved

Description

In the constructor, the input variables `addr` should not be zero address. If a zero address is set to the `_registry), it can never be changed.

Recommendation

We advise the client to check that the aforementioned variables are not zero address.

Alleviation

[APE]: We didn't check that the address is not zero during deployment because we deploy the contracts using a script and there is no risk that that address can be zero. This is intentional to reduce contract size.

RUA-02 | Redundant code

Category	Severity	Location	Status
Logical Issue	Informational	registry/RegistryUser.sol: 13	Resolved

Description

The `Ownable` has variable `_owner` already. Why add the state variable `_owner`?

Recommendation

Consider deleting it if it is useless.

Alleviation

[APE]: It remained from a previous version not extending Ownable. Fixed at https://github.com/ape-market/ape-market-core/pull/89/files#diff-6dd75b6780f441481
73fc24b81485bd7cebacd3cb3497681600ddbd4645f2475R13

SAF-01 | Incorrect judgment condition

Category	Severity	Location	Status
Logical Issue	Minor	nft/SANFTManager.sol: 195	Resolved

Description

When the variable `counter` is 0, the `nft` tokens can not be merged but the code is completely opposite.

Recommendation

We suggest that the condition can be modified like this: `counter <= 1`.

Alleviation

[APE]: Thanks for that, it made us realize a correlated issue and we changed the code. We realized that we were minting a new SANFT even if the buyer was withdrawing everything and emptying the Smart Agreement. Adding a check in SANFTManager._createNewToken for that fixes the issues and simplifies SANFTManager.areMergeable because an SA cannot have a remainingAmount = 0 anymore. Look at https://github.com/ape-market/ape-market-core/pull/91 for the changes

SAF-02 | The remaining amount of the `futureToken` did not modified

Category	Severity	Location	Status
Logical Issue	Minor	nft/SANFTManager.sol: 299	Resolved

Description

The function `swap` forget to update the remaining amount of the sale whose id is `futureTokenSaleId`.

Recommendation

Please check the update of the remaining amount of the `futureTokenSaleId` sale is whether necessary or not. If it is necessary, please consider adding to the code.

Alleviation

[APE]: Short answer: yes, it is what we want. Long answer: If a sale has a futureTokenSaleId, it is not a regular sale, just a swap following a previous futureToken sale. Any user with tokens from the previous sale, identified by futureTokenSaleId, can swap the future token they purchased into the actual token in this sale. The remaining amount of futureTokenSaleId is managed when futureTokenSaleId was going on.

SAF-03 | No upper limit for fee range

Category	Severity	Location	Status
Logical Issue	Medium	nft/SANFTManager.sol: 70	Resolved

Description

The variable `feePoints` is set by the sellers. So it should be within a reasonable range, there should add a check before setting the new value.

Recommendation

Make sure the `feePoints` will be a reasonable value.

Alleviation

[APE]: The `feePoints` represent how much the seller pays Ape for making the sale. The value is set up by the dApp, not by the seller, and an Ape operator has to approve the sale, i.e., the sale setup. Only later, the seller can deploy that sale which will be reverted if any parameter is not consistent with the approved parameters.

APE team removed function `updatePayments`. The change was supplied in the pull request(https://github.com/ape-market/ape-market-core/pull/95/files).

SAM-01 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Logical Issue	Minor	sale/Sale.sol: 62, 53	Resolved

Description

The target `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

Recommendation

It is recommended to use `SafeERC20` or makes sure that the value returned from 'transferFrom()' is checked.

Alleviation

[APE]: Fixed in https://github.com/ape-market/ape-market-core/pull/91

SDA-01 | Possible Incorrect Calculation Formula

Category	Severity	Location	Status
Mathematical Operations	Minor	sale/SaleData.sol: 193	Resolved

Description

The result of `uint256(setup.tokenFeePoints).div(10000)` may always be 1.

```
192  require(
193    tokensAmount <= uint256(setup.remainingAmount).div(1 +
uint256(setup.tokenFeePoints).div(10000)),
194    "SaleData: Not enough tokens available"
195  );
```

Recommendation

Please refer the following formula:

```
uint256(setup.remainingAmount).mul(10000).div(10000 + uint256(setup.tokenFeePoints))
```

Alleviation

[APE]: This is correct. tokenFeePoint is the amount of fee charged in base points (1/10000). For example, the 3% of the fee, is equal to 300 points. It is this way because Solidity does not allow to set, for example, a floating fee like 0.25%.

```
  10100.mul(10000).div(10000 + 100) = 10000
```

Consider that tokenFeePoint can be zero. It means that Ape agreed to let the seller not pay any fee.

[CertiK]: What is the range of `setup.tokenFeePoints`? Given Solidity does not allow to set a floating value, the result of `setup.tokenFeePoints.div(10000)` may be zero forever.

[APE]: APE team changed the formula as follows code.

```
192  uint256 feeOnRemainingAmount =
uint256(setup.remainingAmount).mul(setup.tokenFeePoints).div(10000);
193  require(
194    tokensAmount <= uint256(setup.remainingAmount).sub(feeOnRemainingAmount),
```

```
195    "SaleData: Not enough tokens available"
196  );
```

The change was supplied in the pull request(https://github.com/ape-market/ape-market-core/pull/92/files).

SDA-02 | Redundant ternary expression

Category	Severity	Location	Status
Gas Optimization	Informational	sale/SaleData.sol: 223	Resolved

Description

There are no differences between `vested == 100 ? 0 : uint256(fullAmount).mul(100 - vested).div(100)` and `uint256(fullAmount).mul(100 - vested).div(100)`

Recommendation

Consider using `uint256(fullAmount).mul(100 - vested).div(100)`.

Alleviation

[APE]: Fixed in https://github.com/ape-market/ape-market-core/pull/91

SDA-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	sale/SaleData.sol: 77	Partially Resolved

Description

In the contract `SaleData`, the role `owner` has the authority over the following function:

- `updateApeWallet()`: update the wallet which can accept fees generated in the transaction.

Any compromise to the `owner` account may allow the hacker to take advantage of this and modify the address of the ape wallet to receive fees.

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[APE]: We are aware of that. Will replace the owner with a contract controlled by multi-sig before the final report. And we will introduce the DAO mechanism right after we conduct the public sale. This will ensure that there is governance participation from retail investors and not only early presale investors.

Please refer to the following web for details.

- https://ape-market.gitbook.io/untitled/IyW8vBj5Vv6T5k5I4LVZ/

The APE team added a multi-signature owner in the pull request(https://github.com/ape-market/ape-market-core/pull/96).

SDB-01 | State variable never used

Category	Severity	Location	Status
Logical Issue	Informational	sale/SaleDB.sol: 21	Resolved

Description

State variable `_valuesInEscrow` was never used.

Recommendation

Consider removing the state variable `_valuesInEscrow` if it is useless.

Alleviation

[APE]: It remained from an old version. Fixed at https://github.com/ape-market/ape-market-core/pull/89/files#diff-67149a8f02de01715 c38c0a304c98f4895dbdb92393f6135025fc982bb22ae20L21

SFA-01 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Major	sale/SaleFactory.sol: 73	Resolved

Description

1. The `setUp.tokenFeePoints` is used to calculate the fees in the transaction in sales. However, the value of this variable is set by the seller. So, the value may be very big and cause the investors' loss.
2. The `setUp.remainingAmount` means the amount of `sellingToken` that can be sold. It can be set to any value by the seller before passing to the function `newSale()`.

Recommendation

1. Add checks on the `setUp.tokenFeePoints` in the function `newSale()` to make sure the value is in a reasonable range.
2. Add initialization of the `setUp.remainingAmount`.

Alleviation

[APE]: Point 1. The value is set up by `APE` and approved by `APE` operator. The seller does not have control on it. Point 2. The remaining amount is not set by the seller. The seller (actually, the operator), sets the initial value to 0 (look at this test where there is an example of setup https://github.com/ape-market/ape-market-core/blob/main/test/Sale.test.js#L76). Later, during the creation of the sale, the actual initial value is calculated in the function setLaunchOrExtension in SaleData, at https://github.com/ape-market/ape-market-core/blob/main/contracts/sale/SaleData.sol#L155

SSH-01 | Potential redundant contract

Category	Severity	Location	Status
Logical Issue	Informational	sale/SaleSetupHasher.sol: 2	Resolved

Description

The functions contract `SaleSetupHasher` provided are all included in `SaleLib`.

Recommendation

We recommend removing it if it is useless.

Alleviation

[APE]: The dApp queries that contract to pack the vesting schedule, check the vested percentage, etc. We could have mirrored the code in Javascript, but to avoid risks we preferred to let the solidity library do the calculations all the time. Unfortunately, we can not query the library SaleLib, so we made that contract as an interface to be used by the dApp.

TRA-01 | Lack of Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Informational	sale/TokenRegistry.sol: 39	Resolved

Description

In the constructor, the input variables `addr` should not be zero address. If a zero address is set to the `_registry), it can never be changed.

Recommendation

We advise the client to check that the aforementioned variables are not zero address.

Alleviation

[APE]: We didn't check that the address is not zero during deployment because we deploy the contracts using a script and there is no risk that that address can be zero. This is intentional to reduce contract size.

Appendix

Finding Categories

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.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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.

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

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.

About

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.