Security Assessment

BUDO

Sept 16th, 2021

Table of Contents

Summary

This report has been prepared for Budo, Inc. to discover issues and vulnerabilities in the source code of the BUDO 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	BUDO
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/BUDO2Game/budo-game/tree/main
Commit	97478937424c9a31fb640d63cebe408c603684f2 d4efc11cc8d03080657facd44afefd426697aa0b

Audit Summary

Delivery Date	Sept 16, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

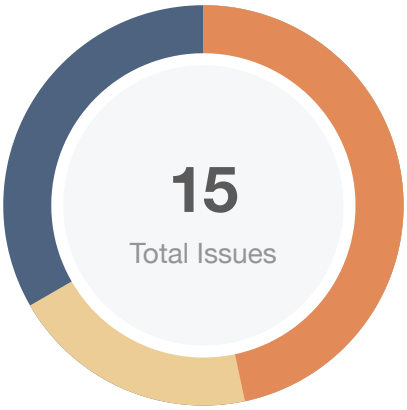
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	7	0	0	7	0	0
Medium	0	0	0	0	0	0
Minor	3	0	0	2	0	1
Informational	5	0	0	1	2	2
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
GBB	GameBlindBox.sol	f929e15d205102b94abc5dfa82725c9c70dc9ec82a18572f1182aa7ddde80095
GCC	GameCard.sol	4d709d07e93fa1a2d05de8b1fff12088fd547c6c5748e19739089914be61d6de
GCS	GameCardStruct.sol	f3f729b019e9816f3b574a5d3337292f93b932ed909c2bc7a5b1ca468c4952dd
GFC	GameFactory.sol	811f09e0790f657a8d5fe119590e805faa22bc5f5a69c39a222edbc137c626b2
GFK	GameFarmingChef.sol	08820e7e944e96dd10050c70b2f9427127b58b6176b82a677cf640d2e156d472
ICC	ICard.sol	431d87808ce59377c250ec54660b27ddea40c9eb6faf25a6b4311b5364a5598c
RGC	RandomGenerator.sol	10c35a2351a45b7d0b3e7b1523cf7525b93c73d13ccb4fa6c857b6405f3a6585

Findings



Critical	**0** (0.00%)
Major	**7** (46.67%)
Medium	**0** (0.00%)
Minor	**3** (20.00%)
Informational	**5** (33.33%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Unknown Imported Source Files and Function	Volatile Code	Informational	Acknowledged
GBB-01	Limit the Execution of Function `safeMint`	**Centralization / Privilege, Logical Issue**	**Major**	Acknowledged
GCC-01	Limit the Execution of Functions	**Logical Issue, Centralization / Privilege**	**Major**	Acknowledged
GCC-02	Lack Of Clearing `cardExp`	Logical Issue	Minor	Resolved
GCC-03	The Purpose of `cardExp`	Logical Issue	Informational	Partially Resolved
GCS-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
GFC-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
GFC-02	3rd party dependencies	Control Flow	Minor	Acknowledged
GFC-03	Centralized risk in `treasury`	**Centralization / Privilege**	**Major**	Acknowledged
GFC-04	Improper Usage of public and external type	Gas Optimization	Informational	Resolved
GFC-05	Risk For Weak Randomness	Volatile Code	Minor	Acknowledged
GFC-06	The Purpose of `cardExp`	Logical Issue	Informational	Partially Resolved
GFK-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
GFK-02	Centralized risk in `treasury`	**Centralization / Privilege**	**Major**	Acknowledged

ID	Title	Category	Severity	Status
GFK-03	Variable could be declared as `constant`	Gas Optimization	Informational	Resolved

GLOBAL-01 | Unknown Imported Source Files and Function

Category	Severity	Location	Status
Volatile Code	Informational	Global	Acknowledged

Description

The imported source files:

1. `../libs/math/SafeMath.sol`
2. `../libs/token/HRC20/IHRC20.sol`
3. `../libs/token/HRC20/SafeHRC20.sol`
4. `../libs/access/Ownable.sol`
5. `../libs/token/HRC721/IHRC721Receiver.sol`
6. `../libs/utils/Counters.sol`
7. `../libs/token/HRC721/extensions/HRC721Enumerable.sol`

are unknown.

The implementation of the function `_exists` is unknown.

Alleviation

The development team replied that all imported files are standard libs from public repository.

GBB-01 | Limit the Execution of Function `safeMint`

Category	Severity	Location	Status
Centralization / Privilege, Logical Issue	**Major**	GameBlindBox.sol: 27	Acknowledged

Description

The owner account can mint nft to anyone at any time by the function `safeMint`. Any compromise to the owner account may allow the hacker to take advantage of this function and eventually damage the contract.

Recommendation

Consider refactoring the code to make the function `safeMint` only be called by the contract `GameFactory`.

Alleviation

The development replied that the owner of GameBlindBox will be `GameFactory`, and no `BlindBox` will be minted before ownership transferred.

GCC-01 | Limit the Execution of Functions

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	**Major**	GameCard.sol: 33~53	Acknowledged

Description

The owner account can mint nft to anyone at any time by the function `safeMint`. The owner account can update activateBlock by the function `activateCard`. The owner account can update cardExp by the function `levelUp`. Any compromise to the owner account may allow the hacker to take advantage of this function and eventually damage the contract.

Recommendation

Consider refactoring the code to make the function `safeMint`, `activateCard` and `levelUp` only be called by the contract `GameFactory`.

Alleviation

The development team replied that the owner of `GameCard` will be `GameFactory`, and no `GameCard` will be minted before ownership transferred, and the activateCard will only be called by `GameFactory`.

GCC-02 | Lack Of Clearing `cardExp`

Category	Severity	Location	Status
Logical Issue	Minor	GameCard.sol: 116~124	Resolved

Description

In the function `_burn`, the `cardExp` of token is not cleared.

Recommendation

Consider clearing the `cardExp` of token:

```
function _burn(uint256 tokenId) internal virtual override {
        super._burn(tokenId);

        if (bytes(_tokenURIs[tokenId]).length != 0) {
            delete _tokenURIs[tokenId];
        }
        activateBlock[tokenId] = 0;
        cardMetas[tokenId] = 0;
        cardExp[tokenId] = 0;
    }
```

Alleviation

The development team heeded our advice and resolved this issue in commit 72bcc9a8d6f5ec69c43b2ecdfb653737ee0943fa.

GCC-03 | The Purpose of `cardExp`

Category	Severity	Location	Status
Logical Issue	Informational	GameCard.sol: 50	Partially Resolved

Description

1. The function `levelUp(uint256 tokenId, uint256 exp)` in the contract `GameCard.sol` only accumulates the value of `exp` instead of increasing the level of a card.
2. What is the purpose of the `cardExp`? It is not taken into account in the function `calFarmingPoints` of the contract `GameFactory.sol`, however, its increment needs burning of some `GameCard` or `ICard`.

Alleviation

`[BUDO team]`: The function levelUp(uint256 tokenId, uint256 exp) in the contract GameCard.sol only accumulates the value of exp instead of increasing the level of a card. The level of card is computed by the cardExp in the game. We make the design clean so that the contract does not manage card levels, but only cardExp.

What is the purpose of the cardExp? It is not taken into account in the function calFarmingPoints of the contract GameFactory.sol, however, its increment needs burning of some GameCard or ICard. cardExp means the total experience points of a card. It is not used in calFarmingPoints because the card experience is not related with farming at all. It is used in battle scenarios.

GCS-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	GameCardStruct.sol: 131~136, 151~154, 164~171, 190~196, 227~229, 267	Acknowledged

Description

In the contract `GameCardStruct`, the role `admin` has the authority over the following functions:

- `pushBox(uint256 boxId, uint256 price, uint256 totalSupply, string memory uri)`
- `updateBoxPrice(uint256 boxId, uint256 price)`
- `pushFamily(uint256 familyId, uint256 totalMembers, uint256 dinnerPoolBonus, uint256 farmingBonus, uint256 reserveBonus1, uint256 reserveBonus2)`
- `pushSpecial(uint256 specialId, uint256 dinnerPoolBonus, uint256 farmingBonus, uint256 reserveBonus1, uint256 reserveBonus2)`
- `pushCardMetaData(uint256[] memory params, string memory uri)`
- `pushButterCardExp(uint256 butterMetaId, uint256 exp)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- generate a blind box
- update the box price
- push a family metadata
- push a special skill metadata
- push a card metadata to the card pool of a specified blind box
- set a card exp

Recommendation

We advise the client to carefully manage the `admin` 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

The development team replied that they will use multi-sign tech in the future and consider giving the admin to the DAO.

GFC-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	GameFactory.sol: 91, 96, 101, 106, 111, 116, 121, 126, 132~134, 141~143	Acknowledged

Description

In the contract `GameFactory`, the role `owner` has the authority over the following functions:

- `setAdmin(address admin_)`
- `setTreasury(address treasury_)`
- `transferCardTokenOwner(address newCardTokenOwner)`
- `transferBoxTokenOwner(address newBoxTokenOwner)`

the role `admin` has the authority over the following functions:

- `setPoolAddress(address poolAddress_)`
- `setBurnRate(uint256 burnRate_)`
- `setTreasuryRate(uint256 treasuryRate_)`
- `setDiscount(uint256 discount_)`
- `setMaxCardSlots(uint256 maxCardSlots_)`
- `setExpRatio(uint256 expRatio_)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- set admin
- set treasury address
- change card token owner
- change box token owner

Any compromise to the `admin` account may allow the hacker to take advantage of this and do the following:

- set pool address
- set burn rate
- set treasure rate
- set discount

- set max card slots
- set exp ratio

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

The development team replied that they will use multi-sign tech in the future and consider giving the admin to the DAO.

GFC-02 | 3rd party dependencies

Category	Severity	Location	Status
Control Flow	Minor	GameFactory.sol: 65~67	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `EACAggregatorProxy` protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume their functional correctness. However, in the real world, 3rd parties may be compromised that led to assets being lost or stolen.

```
priceFeedBTC = AggregatorV3Interface(0xD5c40f5144848Bd4EF08a9605d860e727b991513);
priceFeedHT = AggregatorV3Interface(0x8EC213E7191488C7873cEC6daC8e97cdbAdb7B35);
priceFeedETH = AggregatorV3Interface(0x5Fa530068e0F5046479c588775c157930EF0Dff0);
```

Alleviation

The development replied that the prices are only used to generate a random number, and they are not the whole factors, other factors participates too.

GFC-03 | Centralized risk in `treasury`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	GameFactory.sol: 200	Acknowledged

Description

```
//GameFarmingChef
function unlockSlot() external {
    ...
    token.safeTransfer(treasury, treasuryFee);
}
```

```
//GameFactory.sol
function buyBlindBox(
    uint256 boxId,
    uint256 amount
) external {
...
money.safeTransfer(treasury, treasuryFee);
...
}
```

The `unlockSlot` function of contract `DinnerTableChef` call the `token.safeTransfer` and the function `buyBlindBox` of contract `GameFactory` call the `money.safeTransfer` with the `to` address specified as `treasury`. As a result, over time the `treasury` address will accumulate a significant portion of CAKE tokens. If the `treasury` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

The development team replied that they will use multi-sign tech in the future and consider giving the admin to the DAO.

GFC-04 | Improper Usage of public and external type

Category	Severity	Location	Status
Gas Optimization	Informational	GameFactory.sol: 381, 392	Resolved

Description

public functions that are never called by the contract could be declared external. When the inputs are arrays external functions are more efficient than “public” functions.

Examples:

```
function expectedExp(uint256[] memory foodIds) public
function butterExpectedExp(uint256[] memory butterCards) public
```

Recommendation

Consider using the external attribute for functions never called from the contract.

Alleviation

The development team heeded our advice and resolved this issue in commit d4efc11cc8d03080657facd44afefd426697aa0b.

GFC-05 | Risk For Weak Randomness

Category	Severity	Location	Status
Volatile Code	Minor	GameFactory.sol: 235	Acknowledged

Description

A self-defined function is used to generate the random number.

Recommendation

Consider mixing a seed value based on the trusted 3rd party random service.

Alleviation

The development team replied that chain-link is not supported on heco, they will use chain-link after it is supported.

GFC-06 | The Purpose of `cardExp`

Category	Severity	Location	Status
Logical Issue	Informational	GameFactory.sol: 338	Partially Resolved

Description

1. The function `levelUp(uint256 tokenId, uint256 exp)` in the contract `GameCard.sol` only accumulates the value of `exp` instead of increasing the level of a card.
2. What is the purpose of the `cardExp`? It is not taken into account in the function `calFarmingPoints` of the contract `GameFactory.sol`, however, its increment needs burning of some `GameCard` or `ICard`.

Alleviation

`[BUDO team]`: The function levelUp(uint256 tokenId, uint256 exp) in the contract GameCard.sol only accumulates the value of exp instead of increasing the level of a card. The level of card is computed by the cardExp in the game. We make the design clean so that the contract does not manage card levels, but only cardExp.

What is the purpose of the cardExp? It is not taken into account in the function calFarmingPoints of the contract GameFactory.sol, however, its increment needs burning of some GameCard or ICard. cardExp means the total experience points of a card. It is not used in calFarmingPoints because the card experience is not related with farming at all. It is used in battle scenarios.

GFK-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	GameFarmingChef.sol: 395~399, 163, 175, 169, 181, 186, 197, 202, 208	Acknowledged

Description

In the contract `GameFarmingChef`, the role `owner` has the authority over the following functions:

- `stopRewardAndEmergencyWithdrawAllButter()`
- `setTreasury(address treasury_)`
- `setAdmin(address admin_)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- set `rewardPerBlock` as 0 and transfer pending tokens to the owner
- set treasury address
- set admin address

In the contract `GameFarmingChef`, the role `admin` has the authority over the following functions:

- `setOperator(address operator_)`
- `setDefaultSlotPrice(uint256 defaultUnlockSlotPrice_)`
- `setSlotPrice(uint256[] memory priceArray)`
- `setSlotPriceAt(uint256 pos, uint256 price)`
- `setBurnRate(uint256 burnRate_)`
- `setTreasuryRate(uint256 treasuryRate_)`

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

The development team replied that they will use multi-sign tech in the future and consider giving the admin to the DAO.

GFK-02 | Centralized risk in `treasury`

Category	Severity	Location	Status
Centralization / Privilege	Major	GameFarmingChef.sol: 327	Acknowledged

Description

```
//GameFarmingChef
function unlockSlot() external {
    ...
    token.safeTransfer(treasury, treasuryFee);
}
```

```
//GameFactory.sol
function buyBlindBox(
    uint256 boxId,
    uint256 amount
) external {
...
money.safeTransfer(treasury, treasuryFee);
...
}
```

The `unlockSlot` function of contract `DinnerTableChef` call the `token.safeTransfer` and the function `buyBlindBox` of contract `GameFactory` call the `money.safeTransfer` with the `to` address specified as `treasury`. As a result, over time the `treasury` address will accumulate a significant portion of CAKE tokens. If the `treasury` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;

- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

The development team replied that they will use multi-sign tech in the future and consider giving the admin to the DAO.

GFK-03 | Variable could be declared as `constant`

Category	Severity	Location	Status
Gas Optimization	Informational	GameFarmingChef.sol: 65	Resolved

Description

The variable `blockPerDay` could be declared as `constant` since this state variable is never to be changed.

Recommendation

We recommend declaring those variables as `constant`.

```
uint256 constant public blockPerDay = 28800;
```

Alleviation

The development team heeded our advice and resolved this issue in commit 72bcc9a8d6f5ec69c43b2ecdfb653737ee0943fa.

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.

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.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

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

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.