CERTIK

ShopinX - Audit

Security Assessment

CertiK Assessed on Oct 30th, 2026





CertiK Assessed on Oct 30th, 2026

ShopinX - Audit

The security assessment was prepared by CertiK.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Vesting	EVM Compatible	Manual Review, Static Analysis

LANGUAGE	TIMELINE
Solidity	Preliminary comments published on 10/29/2026 Final report published on 10/30/2026

Vulnerability Summary



7	1	0	6	0
Total Findings	Resolved	Partially Resolved	Acknowledged	Declined

2	Centralization	1 Disabled, 1 Acknowledged	Centralization findings highlight privileged roles & functions and their capabilities, or instances where the project takes custody of users' assets.
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.
0	High		High risks may include logical errors that, under specific circumstances, could result in fund losses or loss of project control.
0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
0	Low		Low 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.
5	Informational	5 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 | SHOPINX - AUDIT

Audit Summary

Executive Summary

Vulnerability Summary

Codebase

Audit Scope

Approach & Methods

Findings

SHA-01 : Initial Token Distribution

SHA-02 : Centralization Risks In ShopinXToken.Sol

SHA-04 : Empty Batches Incorrectly Revert With `BatchTooLarge`

SHA-05 : Distributor Can Bypass The Distribution Cutoff Through ERC20 Transfers

SHA-06 : `batchTransfer()` Permits False Distribution Events For The Distributor Itself

SHA-07 : Distribution Finalization Flag Remains False After Deadline

SHA-08 : Standard `balanceOf` Overstates The Amount Users Can Sell Through PancakeSwap

Appendix

Disclaimer

CODEBASE | SHOPINX - AUDIT

Repository

https://bscscan.com/address/0x4f0ED1f48BEaa4797a62202A21804CaDc2170D6B#readContract

AUDIT SCOPE | SHOPINX - AUDIT

CertiKProject/certik-audit-projects

ShopinXToken.sol

APPROACH & METHODS | SHOPINX - AUDIT

This audit was conducted for ShopinX to evaluate the security and correctness of the smart contracts associated with the ShopinX - Audit project. The assessment included a comprehensive review of the in-scope smart contracts. The audit was performed using a combination of Manual Review and Static Analysis.

The review process emphasized the following areas:

- Architecture review and threat modeling to understand systemic risks and identify design-level flaws.
- Identification of vulnerabilities through both common and edge-case attack vectors.
- Manual verification of contract logic to ensure alignment with intended design and business requirements.
- Assessment of code quality and maintainability, including adherence to current best practices and industry standards.

The audit resulted in findings categorized across multiple severity levels, from informational to critical. To enhance the project's security and long-term robustness, we recommend addressing the identified issues and considering the following general improvements:

- Improve code readability and maintainability by adopting a clean architectural pattern and modular design.
- Strengthen testing coverage, including unit and integration tests for key functionalities and edge cases.
- Maintain meaningful inline comments and documentations.
- Implement clear and transparent documentation for privileged roles and sensitive protocol operations.
- Regularly review and simulate contract behavior against newly emerging attack vectors.

FINDINGS | SHOPINX - AUDIT



7	0	2	0	0	0	5
Total Findings	Critical	Centralization	High	Medium	Low	Informational

This report has been prepared for ShopinX to identify potential vulnerabilities and security issues within the reviewed codebase. During the course of the audit, a total of 7 issues were identified. Leveraging a combination of Manual Review & Static Analysis the following findings were uncovered:

ID	Title	Category	Severity	Status
SHA-01	**Initial Token Distribution**	**Centralization**	**Centralization**	**Acknowledged**
SHA-02	**Centralization Risks In ShopinXToken.Sol**	**Centralization**	**Centralization**	**Disabled**
SHA-04	Empty Batches Incorrectly Revert With `BatchTooLarge`	Coding Issue	Informational	Acknowledged
SHA-05	Distributor Can Bypass The Distribution Cutoff Through ERC20 Transfers	Logical Issue	Informational	Acknowledged
SHA-06	`batchTransfer()` Permits False Distribution Events For The Distributor Itself	Coding Issue	Informational	Acknowledged
SHA-07	Distribution Finalization Flag Remains False After Deadline	Logical Issue	Informational	Acknowledged
SHA-08	Standard `balanceOf` Overstates The Amount Users Can Sell Through PancakeSwap	Logical Issue	Informational	Acknowledged

SHA-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization	**Centralization**	**ShopinXToken.sol: 114~115**	**Acknowledged**

Description

All of the ShopinX tokens are sent to the `distributor_` address. This is a centralization risk because the `distributor_` can distribute tokens without obtaining the consensus of the community. Any compromise to this address may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize the project team with a third-party KYC provider to create greater accountability.

Alleviation

[ShopinX, 10/29/2026]: The team acknowledged this issue. The distribution plan is published at https://shopinx.io/migration. The undistributed balance is held by a Ledger hardware wallet: https://bscscan.com/address/0xd7464e43cbf16Db762A650646FA9ddFC343eb5bA (no multisig at this stage).

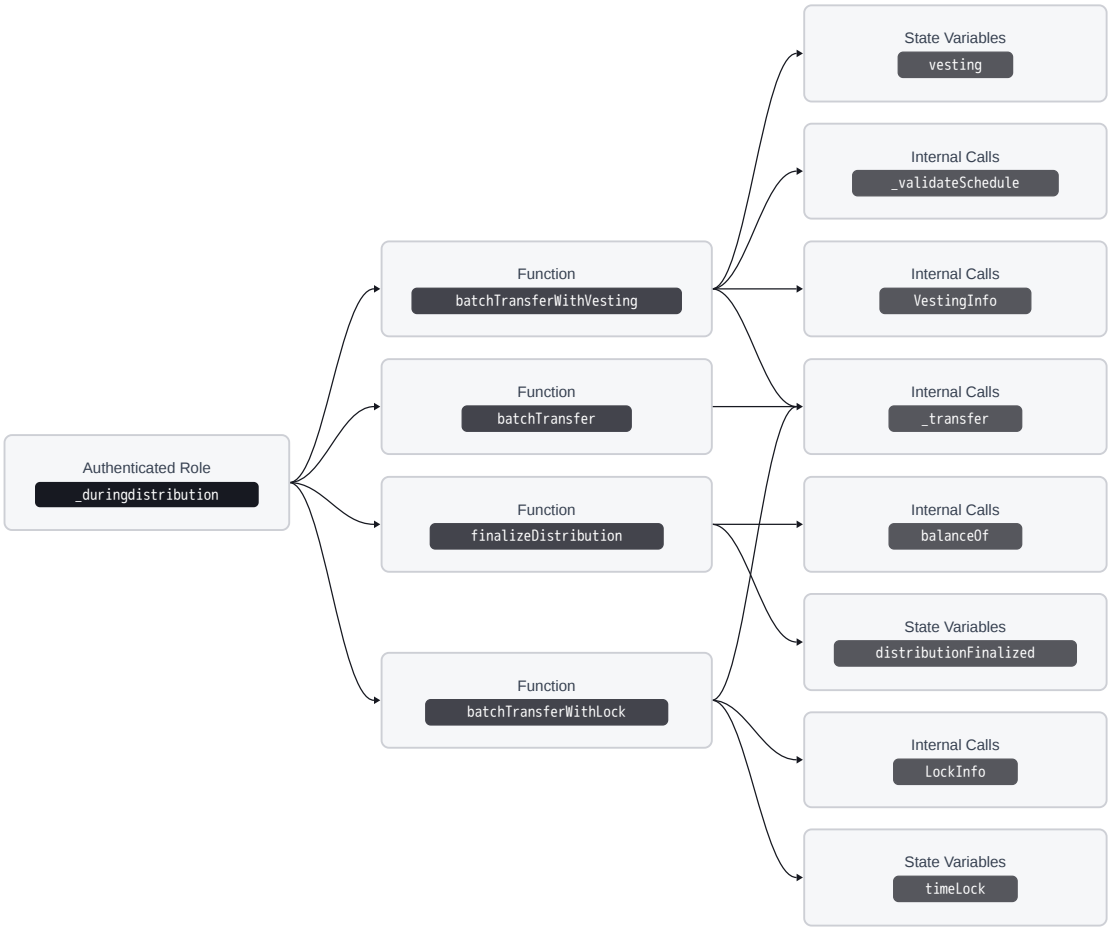
[CertiK, 10/29/2026]: It is suggested to implement the aforementioned methods to avoid centralized failure. Also, CertiK strongly encourages the project team to periodically revisit the private key security management of all addresses related to centralized roles.

SHA-02 | Centralization Risks In ShopinXToken.Sol

Category	Severity	Location	Status
Centralization	**Centralization**	**ShopinXToken.sol: 131, 149, 179, 199**	**Disabled**

Description

In the contract `ShopinXToken` the role `distributor` has authority over the functions shown in the diagram below. Any compromise to the `distributor` account may allow the hacker to take advantage of this authority.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

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

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

Long Term:

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

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

Permanent:

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

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

Alleviation

[ShopinX, 10/30/2026]:

The `distributor` role expired at the distribution deadline (29 Oct 2026, 13:49:28 UTC). `isDistributionActive()` returns false, so no privileged function can be called by anyone anymore. The contract never had an owner, so there is no separate renounce transaction; the deadline itself closed the role permanently.

[CertiK, 10/30/2026]:

Given the current on-chain state of the ShopinX contract, all the privileged functions identified in this finding—`batchTransfer()`, `batchTransferWithVesting()`, `batchTransferWithLock()`, and `finalizeDistribution()`—cannot be successfully invoked.

SHA-04 | Empty Batches Incorrectly Revert With `BatchTooLarge`

Category	Severity	Location	Status
Coding Issue	Informational	ShopinXToken.sol: 137, 154, 182	Acknowledged

Description

All three batch distribution functions use `BatchTooLarge` for both empty and oversized inputs. For empty inputs, the error incorrectly indicates that the batch exceeds the maximum size, potentially misleading integrators diagnosing rejected transactions.

- In `batchTransfer`, an empty pair of input arrays reverts with `BatchTooLarge`, although the length is zero rather than greater than `MAX_BATCH_SIZE`.
- In `batchTransferWithVesting`, an empty `grants` array reverts with `BatchTooLarge`, although no upper limit has been exceeded.
- In `batchTransferWithLock`, an empty `grants` array reverts with `BatchTooLarge`, although no upper limit has been exceeded.

Recommendation

Use a separate `EmptyBatch` error for zero-length inputs and retain `BatchTooLarge` for inputs exceeding `MAX_BATCH_SIZE`, or rename the shared error to `InvalidBatchSize`.

Alleviation

[ShopinX, 10/29/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement. Moreover, the contract is deployed and immutable, so these will be addressed in any future version.

SHA-05 Distributor Can Bypass The Distribution Cutoff Through ERC20 Transfers

Category	Severity	Location	Status
Logical Issue	Informational	ShopinXToken.sol: 33, 114~115, 118~125, 199~201, 271~283	Acknowledged

Description

The contract intends to permit distributor-originated allocations only during a one-time distribution phase. The constructor assigns the full token supply to `distributor`, while the custom batch distribution functions and `finalizeDistribution()` are protected by `onlyDuringDistribution`.

ShopinXToken.sol

```
114         distributionDeadline = block.timestamp + DISTRIBUTION_WINDOW;
115         _mint(distributor_, TOTAL_SUPPLY);
```

```
118     modifier onlyDuringDistribution() {
119         if (msg.sender != distributor) revert NotDistributor();
120         if (!isDistributionActive()) revert DistributionClosed();
121         _;
122     }
```

The inherited ERC20 entry points do not use this modifier. They call `_transfer`, which reaches the contract's `_update()` implementation, but `_update()` only enforces vesting and time-lock balances. The distributor has no vesting or time-lock metadata, so its remaining balance is treated as fully unlocked.

OpenZeppelin ERC20.sol

```
 99     function transfer(address to, uint256 value) public virtual returns (bool)
{
100         address owner = _msgSender();
101         _transfer(owner, to, value);
102         return true;
103     }
```

```
142     function transferFrom(address from, address to, uint256 value) public
virtual returns (bool) {
143         address spender = _msgSender();
144         _spendAllowance(from, spender, value);
145         _transfer(from, to, value);
146         return true;
147     }
```

Consequently, after the deadline passes or `finalizeDistribution()` sets `distributionFinalized`, the distributor can still move undistributed tokens through `transfer()`. A spender that received an allowance can also move those tokens later through `transferFrom()`. These transfers do not create vesting or time-lock metadata and therefore allow post-closure allocations without the controls applied by the custom distribution functions.

The issue primarily weakens the documented cutoff and distribution accounting because it requires action by the distributor or an address that the distributor has approved. It does not independently grant an arbitrary third party access to the distributor's balance.

Recommendation

Enforce the distribution closure policy for every transfer whose source is `distributor`. One option is to add a check in `_update()` that reverts when `from == distributor` and `!isDistributionActive()`, while deciding explicitly whether post-closure burns should also be blocked. Alternatively, escrow, burn, or otherwise account for the remaining distributor balance when finalizing the distribution.

Alleviation

[ShopinX, 10/29/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement. Moreover, the remaining distributor balance is reserved for exchange allocations and pending holder deliveries and will be disclosed transparently.

SHA-06 `batchTransfer()` Permits False Distribution Events For The Distributor Itself

Category	Severity	Location	Status
Coding Issue	Informational	ShopinXToken.sol: 140~143	Acknowledged

Description

The plain batch path rejects only the zero address and zero amounts. It does not reject `to[i] == distributor`, even though the vesting and time-lock batch functions explicitly reject the distributor as a recipient.

When the distributor includes itself, `_transfer(distributor, distributor, amount)` subtracts and re-adds the same balance through OpenZeppelin's shared `_update` path, then `batchTransfer` emits `Distributed(distributor, amount)`. The distributor has no net allocation, but event-based allocation accounting can record a distribution that did not leave the distributor.

Recommendation

Reject the distributor as a plain-batch recipient, for example by checking `to[i] == address(0) || to[i] == distributor` before `_transfer`, or otherwise prevent `Distributed` from being emitted for self-transfers.

Alleviation

[ShopinX, 10/29/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement. Moreover, the contract is deployed and immutable, so these will be addressed in any future version.

SHA-07 | Distribution Finalization Flag Remains False After Deadline

Category	Severity	Location	Status
Logical Issue	Informational	ShopinXToken.sol: 120, 198~202	Acknowledged

Description

The contract uses `distributionDeadline` to close the initial batch-distribution phase, while `distributionFinalized` records only an explicit call to `finalizeDistribution()` . The distribution modifier rejects calls once the deadline has passed:

ShopinXToken.sol

```
118     modifier onlyDuringDistribution() {
119         if (msg.sender != distributor) revert NotDistributor();
120 @>      if (!isDistributionActive()) revert DistributionClosed();
121         _;
122     }
123
124     function isDistributionActive() public view returns (bool) {
125         return !distributionFinalized && block.timestamp <=
distributionDeadline;
126     }
```

However, `finalizeDistribution()` can only be called while that same modifier succeeds:

ShopinXToken.sol

```
198     /// @notice Permanently ends the distribution phase. Irreversible.
199     function finalizeDistribution() external onlyDuringDistribution {
200         distributionFinalized = true;
201         emit DistributionFinalized(block.timestamp, balanceOf(distributor));
202     }
```

If the distributor does not call `finalizeDistribution()` before `distributionDeadline` , the batch-distribution functions remain closed because `block.timestamp <= distributionDeadline` becomes false, but `distributionFinalized` remains `false` permanently and no `DistributionFinalized` event is emitted. This creates an ambiguous terminal state for off-chain consumers that use the public flag or event to determine whether distribution has ended. The condition does not reopen the batch-distribution functions, so the impact is limited to state reporting, monitoring, and integrations that rely on the explicit finalization marker.

Recommendation

Make deadline expiry produce an explicit and unambiguous terminal state.

One option is to add a one-time `finalizeAfterDeadline()` state-synchronization function that sets `distributionFinalized` to `true` and emits `DistributionFinalized` after the deadline, or to expose and standardize a derived `isDistributionClosed()` value that treats either explicit finalization or deadline expiry as closed. Update integrations to use that canonical state instead of relying on `distributionFinalized` alone.

Alleviation

[ShopinX, 10/29/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement. Moreover, the contract is deployed and immutable, so these will be addressed in any future version.

SHA-08 Standard `balanceOf` Overstates The Amount Users Can Sell Through PancakeSwap

Category	Severity	Location	Status
Logical Issue	Informational	ShopinXToken.sol	Acknowledged

Description

The ShopinX token reports an account's full token balance through the standard ERC20 `balanceOf()` function, while its `_update()` hook prevents transfers exceeding the account's currently unlocked balance. The contract exposes `availableToTransfer()` for this purpose, but standard wallets, DEX interfaces, and token integrations generally rely on `balanceOf()`.

For example, an account may have:

```
balanceOf(user)            = 1,000
lockedBalanceOf(user)      =   700
availableToTransfer(user) =   300
```

The user interface may therefore allow the user to sell 1,000 tokens. Although an approval for 1,000 tokens succeeds, PancakeSwap V3's Router will eventually call:

```
transferFrom(user, pool, amountIn)
```

When `amountIn` exceeds 300, the token's `_update()` function reverts with `TransferExceedsUnlocked`. The same issue affects adding liquidity through a position manager or any other integration that transfers tokens from the user.

The ShopinX website currently directs users to trade the token through PancakeSwap V3 in the `SHOPINX/WBNB` pool (official page). Consequently, users with locked allocations may see a balance that is larger than the amount they can actually sell or provide as liquidity. Buy operations generally remain successful because the PancakeSwap pool has no vesting or time-lock metadata.

This does not affect the pool's pricing or reserve accounting and does not allow funds to be stolen. Failed swaps are reverted atomically, but users may incur failed transaction gas and integrations may display misleading maximum trade amounts.

The frontend, SDK, and other integrations should query:

```
availableToTransfer(user)
```

when calculating the maximum transferable or sellable amount, while continuing to use `balanceOf()` for total holdings.

Recommendation

The audit team recommends the custom spendable-balance behavior should also be documented for DEX and wallet integrators.

Alleviation

[ShopinX, 10/29/2026]: The team acknowledged the issue and decided not to implement the recommended change in the current engagement. Moreover, in practice the impact is temporary: • All migration time-locks expire today, 29 Oct 2026 at 13:49:28 UTC. After that, balanceOf equals the transferable balance for every holder. • The only remaining locks are the three treasury wallets under on-chain vesting (Reserve, Marketing, Team), which do not trade on the DEX. • We document availableToTransfer(address) as the correct spendable-balance function on our transparency page, and our own frontend uses it.

APPENDIX | SHOPINX - AUDIT

Finding Categories

Categories	Description
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.

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.

Elevate Your Web3 Journey

CertiK is the largest Web3 security platform combining formal verification with audits and comprehensive security solutions.