



# Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Token	Documentation quality	High	<div></div>
Timeline	2025-06-20 through 2025-06-23	Test quality	High	<div></div>
Language	Solidity	Total Findings	1	<div>Fixed: 1</div>
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review	High severity findings ⓘ	0	
Specification	None	Medium severity findings ⓘ	0	
Source Code	<ul style="list-style-type: none"><li><a href="https://github.com/openledger-dev/openledger">https://github.com/openledger-dev/openledger</a> </li><li><a href="#">#4aab6ac</a> </li></ul>	Low severity findings ⓘ	0	
Auditors	<ul style="list-style-type: none"><li>Hamed Mohammadi Auditing Engineer</li><li>Rabib Islam Senior Auditing Engineer</li></ul>	Undetermined severity findings ⓘ	0	
		Informational findings ⓘ	1	<div>Fixed: 1</div>

# Summary of Findings

The `WOPEN` contract aims to provide wrapped token functionality for the `OPEN` token on the OpenLedger Network, similar to how `WETH` provides wrapped token functionality for `ETH` on Ethereum mainnet. The contract code is taken from the `WETH` code with some adjustments to prevent attacks similar to the `WETH` permit attack. Notably, the contract takes advantage of a more recent Solidity version and uses the `receive()` function instead of a nameless function that existed in prior Solidity versions. In this case, the `receive()` function will only accept calls that have no payload, meaning the `WETH` permit attack will be impossible.

The `GOPEN` token aims to provide voting functionality for holders of the `OPEN` token with the help of OpenZeppelin's `ERC20Votes` contract.

Overall, both contracts are well-written, well-structured, and free of major vulnerabilities.

## Fix-Review Update 2025-06-25:

Repository: `https://github.com/openledger-dev/openledger` Commit: `b6cb12f4977d7f98ed9b43f9c722ec58ec6f7346`  
All fixes have been reviewed up to the referenced commit, and all identified issues have been fully addressed and resolved.

ID	DESCRIPTION	SEVERITY	STATUS
OPE-1	Use <code>Call</code> Instead of <code>transfer</code> for Sending Eth	<ul style="list-style-type: none"><li>Informational ⓘ</li></ul>	Fixed

# Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

## Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
  1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
  2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
  3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
  1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
  2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

Files Included

Repo: https://github.com/openledger-dev/openledger

Included Paths: contracts/WOPEN.sol , contracts/GOPEN.sol

Files Excluded

Files: https://github.com/openledger-dev/openledger/blob/master/contracts/Open.sol

Operational Considerations

The **OPEN** token will act as the native currency in its designated L2 chain.

The **GOPEN** contract uses voting durations expressed as timestamps.

Key Actors And Their Capabilities

Contracts do not have privileged roles defined.

Findings

OPE-1 Use **Call** Instead of **transfer** for Sending Eth • Informational ⓘ **Fixed**

✓

Update

Marked as "Fixed" by the client.

Addressed in: 89fb8f22cdb53b27c72135c7b03cb69baaec73c3 .

File(s) affected: contracts/WOPEN.sol

**Description:** The `transfer()` and `send()` functions forward only 2300 gas to the recipient, a design once widely recommended as a safeguard against reentrancy attacks. However, this fixed gas stipend creates a fragility: changes to EVM gas costs introduced in hard forks can render contracts inoperable. For instance, EIP-1884 increased the gas cost of certain opcodes like `SLOAD`, breaking existing contracts that relied on the assumption that 2300 gas would be sufficient for execution.

In addition, the rise of smart accounts (e.g., ERC-4337-based wallets) makes `transfer()` and `send()` increasingly incompatible. These accounts can use contract logic to handle incoming ETH, which may requires more than 2300 gas to execute. Using `.call{value: amount}()` ensures compatibility with these evolving account architectures and supports the broader adoption of programmable and user-friendly wallet systems.

**Recommendation:** Use `call()` to prevent potential gas issues.

# Auditor Suggestions

## S1 Token Properties Should Be Constants

Fixed



### Update

Marked as "Fixed" by the client.  
Addressed in: `b6cb12f4977d7f98ed9b43f9c722ec58ec6f7346`.

**File(s) affected:** `contracts/WOPEN.sol`

**Description:** The three token properties `name`, `symbol`, and `decimals` of the `WOPEN` contract can be marked as constant to improve code maintainability.

**Recommendation:** Consider marking these properties with the `constant` keyword.

# Definitions

- High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- Undetermined** – The impact of the issue is uncertain.
- Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

# Appendix

### File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

### Files

Repo: `https://github.com/openledger-dev/openledger`

- `923...aea ./contracts/GOPEN.sol`
- `a44...494 ./contracts/WOPEN.sol`

# Toolset

The notes below outline the setup and steps performed in the process of this audit.

## Setup

Tool Setup:

- [Slither](#)  v0.10.0

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

# Automated Analysis

## Slither

We ran Slither, which analyzed 41 contracts using 64 detectors, resulting in 20 findings. Most of them are false positives or out of scope. We have included the valid ones in the report as issues or suggestions.

# Test Suite Results

We were able to install all the required dependencies using the `npm install` command and run the entire test suite using the `npx hardhat test` command.

```
> HardhatEVM: v2.24.2
> network:      hardhat

GOpen Token – Reentrancy Security Test
  Normal Operations
    ✓ Should handle deposit and withdrawal correctly
    ✓ Should maintain correct ETH balance in contract
  Reentrancy Attack Prevention
    ✓ Should prevent reentrancy attacks – no profit extraction (43ms)
    ✓ Should burn tokens before external call (43ms)
    ✓ Should handle multiple reentrancy attempts (39ms)
  Edge Cases
    ✓ Should handle small amounts correctly (39ms)
    ✓ Should reject zero value operations
  State Consistency
    ✓ Should maintain consistent state after attack attempts (55ms)

ERC20Bank Exploit Test
Deploying BaseWETH...
BaseWETH target: 0x67d269191c92Caf3cD7723F116c85e6E9bf55933
Deploying ERC20Bank for BaseWETH...
ERC20Bank for BaseWETH address: 0xE6E340D132b5f46d1e472DebcD681B2aBc16e57E
Calling deposit on BaseWETH...
Deposit successful
User depositing into bank...
User address: 0xf39Fd6e51aad88F6F4ce6aB8827279cFfFb92266
Deploying WOpen...
WOpen address: 0xa82fF9aFd8f496c3d6ac40E2a0F282E47488CFc9
Deploying ERC20Bank for WOpen...
ERC20Bank for WOpen address: 0x1613beB3B2C4f22Ee086B2b38C1476A3cE7f78E8
User WETH balance: 99.0
Attacker bank balance: 99.0
Final user WETH balance: 0.0
Final attacker WETH balance: 99.0
  ✓ BaseWETH should be vulnerable to WETH permit attack
Deploying BaseWETH...
BaseWETH target: 0x851356ae760d987E095750cCeb3bC6014560891C
Deploying ERC20Bank for BaseWETH...
ERC20Bank for BaseWETH address: 0xf5059a5D33d5853360D16C683c16e67980206f36
Calling deposit on BaseWETH...
```

```
Deposit successful
User depositing into bank...
User address: 0xf39Fd6e51aad88F6F4ce6aB8827279cfffFb92266
Deploying WOpen...
WOpen address: 0x4826533B4897376654Bb4d4AD88B7faFD0C98528
Deploying ERC20Bank for WOpen...
ERC20Bank for WOpen address: 0x99bbA657f2BbC93c02D617f8bA121cB8Fc104AcF
User WOpen balance: 0.0
    ✓ WOpen should not be vulnerable to WETH permit attack
Deploying BaseWETH...
BaseWETH target: 0x0E801D84Fa97b50751Dbf25036d067dCf18858bF
Deploying ERC20Bank for BaseWETH...
ERC20Bank for BaseWETH address: 0x8f86403A4DE0BB5791fa46B8e795C547942fE4Cf
Calling deposit on BaseWETH...
Deposit successful
User depositing into bank...
User address: 0xf39Fd6e51aad88F6F4ce6aB8827279cfffFb92266
Deploying WOpen...
WOpen address: 0x809d550fca64d94Bd9F66E60752A544199cfAC3D
Deploying ERC20Bank for WOpen...
ERC20Bank for WOpen address: 0x4c5859f0F772848b2D91F1D83E2Fe57935348029
    ✓ Valid depositWithPermit for BaseWETH
Deploying BaseWETH...
BaseWETH target: 0x1291Be112d480055DaFd8a610b7d1e203891C274
Deploying ERC20Bank for BaseWETH...
ERC20Bank for BaseWETH address: 0x5f3f1dBD7B74C6B46e8c44f98792A1dAf8d69154
Calling deposit on BaseWETH...
Deposit successful
User depositing into bank...
User address: 0xf39Fd6e51aad88F6F4ce6aB8827279cfffFb92266
Deploying WOpen...
WOpen address: 0x2bdCC0de6bE1f7D2ee689a0342D76F52E8EFABa3
Deploying ERC20Bank for WOpen...
ERC20Bank for WOpen address: 0x7969c5eD335650692Bc04293B07F5BF2e7A673C0
    ✓ Withdraw functionality for WOpen
Deploying BaseWETH...
BaseWETH target: 0xFD471836031dc5108809D173A067e8486B9047A3
Deploying ERC20Bank for BaseWETH...
ERC20Bank for BaseWETH address: 0xcbEAF3BDe82155F56486Fb5a1072cb8baAf547cc
Calling deposit on BaseWETH...
Deposit successful
User depositing into bank...
User address: 0xf39Fd6e51aad88F6F4ce6aB8827279cfffFb92266
Deploying WOpen...
WOpen address: 0x922D6956C99E12DFeB3224DEA977D0939758A1Fe
Deploying ERC20Bank for WOpen...
ERC20Bank for WOpen address: 0x5081a39b8A5f0E35a8D959395a630b68B74Dd30f
    ✓ Reentrancy prevention in WOpen

GOpen
Deployment
    ✓ Should set the right name and symbol
    ✓ Should start with zero total supply
Deposits
    ✓ Should mint tokens when depositing ETH
    ✓ Should handle minimum deposit of 1 wei
    ✓ Should not allow zero value deposits
    ✓ Should handle large deposits
Withdrawals
    ✓ Should burn tokens and return ETH when withdrawing
    ✓ Should allow withdrawing exact balance amount
    ✓ Should revert when withdrawing more than balance
    ✓ Should handle reentrancy attempts (41ms)
Gas used for withdrawal: 80486
    ✓ Should measure gas usage for withdrawals
Governance and Voting
    ✓ Should track votes and past votes correctly
    ✓ Should track past total supply correctly
    ✓ Should handle checkpoints correctly
    ✓ Should allow delegation
    ✓ Should simulate governance with multiple users (45ms)
Permit
```

✓ **Should** permit **and** transfer using signature

Receive Function

✓ **Should** mint tokens when receiving ETH **directly**

Custom Errors

✓ **Should** revert with ZeroValueNotAllowed on **zero**-value deposit

✓ **Should** revert with ZeroValueNotAllowed on **zero**-value withdrawal

Overrides

✓ **Should** return the correct **clock** value

✓ **Should** return the correct **CLOCK\_MODE**

✓ **Should** update **balances** correctly via **\_update**

✓ **Should** return the correct nonce for an **address**

GOpen Token Complete Checkpoint Test

Initial State:

Total Supply: **0.0**

Current Time: **1750639059**

Past Total Supply: **0.0**

Deposit **1** ETH:

Deposit completed **at** time: **1750639060**

After **2 blocks**:

User1 **Balance**: **1.0**

Delegate to self:

Delegation completed **at** time: **1750639063**

User1's **delegatee**: **0x70997970C51812dc3A010C7d01b50e0d17dc79C8**

After **2 more blocks**:

User1 Current Votes: **1.0**

Checking past votes at time **1750639067**

User1 Past Votes: **1.0**

Total Supply before withdrawal: **1.0**

Withdrawing **1** ETH:

Withdrawal completed at time: **1750639068**

After withdrawal and **2 blocks**:

User1 Votes after withdrawal: **0.0**

Final checks at time **1750639072**

User1 Past Votes after withdrawal: **0.0**

Checkpoint Information:

Number of checkpoints for User1: **2**

Final Supply State:

Final Total Supply: **0.0**

Final Past Total Supply: **0.0**

✓ should demonstrate all checkpoint and voting scenarios (64ms)

Open

Deployment

✓ Should set the right name and symbol

✓ Should assign the total supply of tokens to the owner

✓ Should have correct total supply

Transactions

✓ Should transfer tokens between accounts

✓ Should fail if sender doesn't have enough tokens

✓ **Should** update allowances on approval

TransferFrom

✓ **Should** transfer tokens using transferFrom

✓ **Should** fail if trying to transferFrom more than allowed

Burning

✓ **Should burn** tokens **and** reduce total supply

✓ **Should** allow users to **burn** their own tokens

✓ **Should** fail if trying to **burn** more tokens than owned

✓ **Should** emit Transfer event on **burn**

WOpen



Deployment

- ✓ **Should** set the right name **and** symbol
- ✓ **Should** start with **zero** total supply

Deposits

- ✓ **Should** mint tokens when depositing ETH
- ✓ **Should** handle minimum deposit of **1** wei

Withdrawals

- ✓ **Should** **burn** tokens **and** return ETH when withdrawing
- ✓ **Should** revert when withdrawing more than **balance**
- ✓ **Should** handle partial withdrawals

Transfers

- ✓ **Should** transfer tokens **between** accounts
- ✓ **Should** fail if sender doesn't **have enough tokens**
- ✓ **Should** **handle self-transfers without checking allowance**
- ✓ **Should** **not decrease allowance when it's** set to maximum (infinite approval)
- ✓ **Should** decrease allowance on transferFrom with finite approval

Allowances

- ✓ **Should** update allowance on approval
- ✓ **Should** not affect allowance on transfer
- ✓ **Should** handle infinite approvals

Total Supply

- ✓ **Should** track total supply correctly through deposits **and** withdrawals

Receive Function

- ✓ **Should** handle Ether sent **directly** to the contract
- ✓ **Should** revert if msg.**data** is not empty

Edge Cases

Transfer Edge Cases

- ✓ **Should** fail transferFrom when allowance is **insufficient**
- ✓ **Should** allow transfer of exact allowance amount
- ✓ **Should** handle **zero** value transfers

Deposit Edge Cases

- ✓ **Should** handle **zero** value deposits
- ✓ **Should** handle **multiple** consecutive deposits

Withdrawal Edge Cases

- ✓ **Should** handle **zero** value withdrawals
- ✓ **Should** handle withdrawal of exact **balance**

75 passing (4s)

# Code Coverage

The test coverage was generated via the `npx hardhat coverage` command. The results show perfect 100% coverage across all categories.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	100	100	100	
GOPEN.sol	100	100	100	100	
Open.sol	100	100	100	100	
WOPEN.sol	100	100	100	100	
All files	100	100	100	100	

# Changelog

- 2025-06-23 - Initial report
- 2025-06-26 - Final report

# About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over \$200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

## Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

## Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

## Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites' owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

## Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.



