Security Review Report
NM-0411-0535 - Worldchain-Operator-Payouts

NETHERMIND
SECURITY

(May 14, 2025)

Contents

1 Executive Summary

This document presents the security review performed by Nethermind Security for the Worldcoin team. The review focuses on the `OperatorPayoutsWorldchain` contract, which is designed to be called by the Operator backend to distribute rewards. These rewards are denominated in USD and are converted to an equivalent amount in WLD tokens for payout.

The audit comprises 78 lines of solidity code. **The audit was performed using** (a) manual analysis of the codebase, (b) automated analysis tools, and (c) creation of test cases.

Along this document, we report 4 points of attention, where one are classified as `Low`, and three are classified as `Informational` and `Best Practices`. The issues are summarized in Fig. 1.

This document is organized as follows. Section 2 presents the files in the scope. Section 3 summarizes the issues. Section 4 presents the system overview. Section 5 discusses the risk rating methodology. Section 6 details the issues. Section 7 discusses the documentation provided by the client for this audit. Section 8 presents the compilation, tests, and automated tests. Section 9 concludes the document.

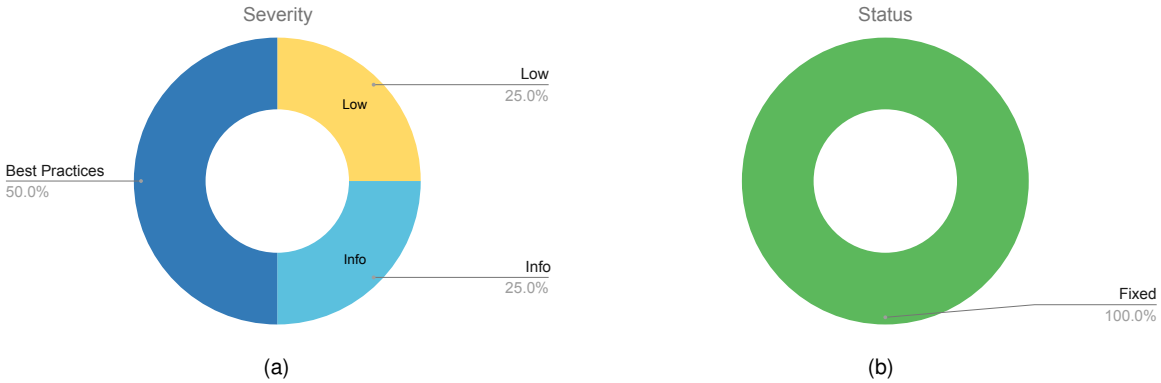

Fig. 1: Distribution of issues: Critical (0), **High** (0), **Medium** (0), **Low** (1), **Undetermined** (0), **Informational** (1), **Best Practices** (2). **Distribution of status: Fixed** (4), **Acknowledged** (0), **Mitigated** (0), **Unresolved** (0)

Summary of the Audit

Audit Type	Security Review
Initial Report	May 09, 2025
Final Report	May 14, 2025
Repositories	helper-contracts
Initial Commit	f3147a836203d3ca99f56debfa88ba91c1ccaade
Final Commit	156e93bee5b08f9ca46ba80f06929ed4fa23c4c0
Documentation	PR description
Documentation Assessment	High
Test Suite Assessment	High

2 Audited Files

	Contract	LoC	Comments	Ratio	Blank	Total
1	OperatorPayoutsWorldchain.sol	78	44	56.4%	28	150
	Total	**78**	**44**	**56.4%**	**28**	**150**

3 Summary of Issues

	Finding	Severity	Update
1	Potential use of stale WLD price in payout calculations	Low	Fixed
2	High `minWldPrice` can lead to zero `tokenAmount`	Info	Fixed
3	Missing event emission in the `setRelayer` function	Best Practices	Fixed
4	Missing validation for zero `dollarAmount`	Best Practices	Fixed

4 System Overview

The `OperatorPayoutsWorldchain` contract enables the transfer of WLD token payouts by an authorized relayer, which is intended to be the Operator backend. The contract accepts a reward amount specified in USD, as a whole number with zero decimals, and converts it to the equivalent WLD amount using the WLD/USD price provided by the Chainlink oracle.

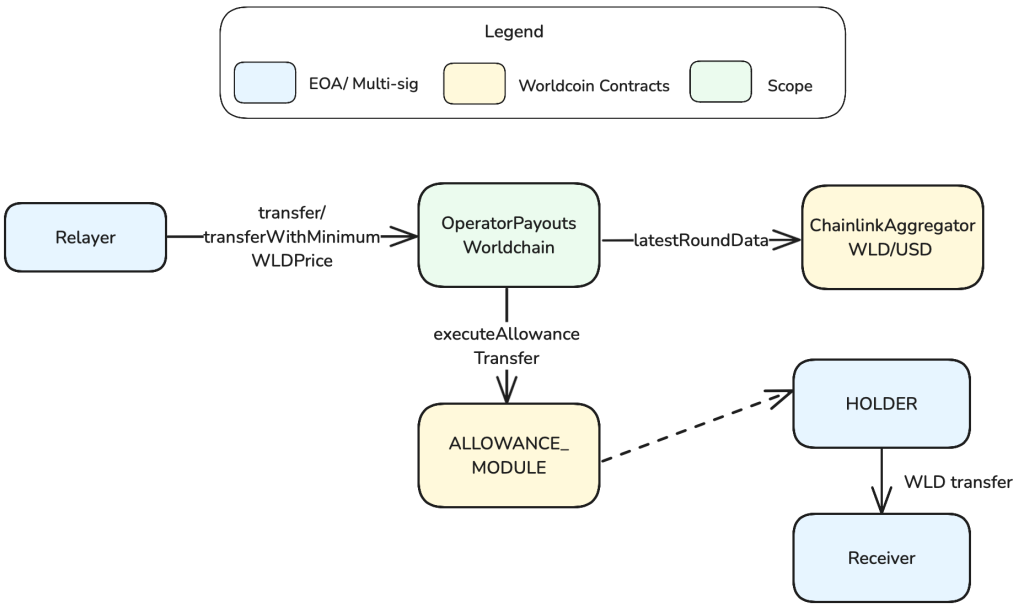

Fig. 2: Worldchain Operator Payouts Overview

The contract exposes two main functions for token transfers:

```
function transfer(address payable to, uint256 dollarAmount) external onlyRelayer
```

This function transfers WLD tokens to the specified recipient address based on the current Chainlink oracle price. Transfers are executed through Worldcoin's allowance module.

```
function transferWithMinimumWLDPrice(
        address payable to,
        uint256 dollarAmount,
        uint256 minWldPrice
) external onlyRelayer
```

This variant allows the relayer to specify a minimum acceptable WLD/USD price. If the oracle price falls below this minimum, the transfer will use the specified minimum price instead, resulting in a lower WLD payout. This function is used as a safeguard in cases of large WLD price drops.

5 Risk Rating Methodology

The risk rating methodology used by Nethermind Security follows the principles established by the OWASP Foundation. The severity of each finding is determined by two factors: **Likelihood** and **Impact**.

Likelihood measures how likely the finding is to be uncovered and exploited by an attacker. This factor will be one of the following values:

a) **High**: The issue is trivial to exploit and has no specific conditions that need to be met;

b) **Medium**: The issue is moderately complex and may have some conditions that need to be met;

c) **Low**: The issue is very complex and requires very specific conditions to be met.

When defining the likelihood of a finding, other factors are also considered. These can include but are not limited to motive, opportunity, exploit accessibility, ease of discovery, and ease of exploit.

Impact is a measure of the damage that may be caused if an attacker exploits the finding. This factor will be one of the following values:

a) **High**: The issue can cause significant damage, such as loss of funds or the protocol entering an unrecoverable state;

b) **Medium**: The issue can cause moderate damage, such as impacts that only affect a small group of users or only a particular part of the protocol;

c) **Low**: The issue can cause little to no damage, such as bugs that are easily recoverable or cause unexpected interactions that cause minor inconveniences.

When defining the impact of a finding, other factors are also considered. These can include but are not limited to Data/state integrity, loss of availability, financial loss, and reputation damage. After defining the likelihood and impact of an issue, the severity can be determined according to the table below.

		Severity Risk		
Impact	**High**	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Info/Best Practices	Low	Medium
	Undetermined	Undetermined	Undetermined	Undetermined
		Low	**Medium**	**High**
		Likelihood		

To address issues that do not fit a High/Medium/Low severity, Nethermind Security also uses three more finding severities: **Informational**, **Best Practices**, and **Undetermined**.

a) **Informational** findings do not pose any risk to the application, but they carry some information that the audit team intends to pass to the client formally;

b) **Best Practice** findings are used when some piece of code does not conform with smart contract development best practices;

c) **Undetermined** findings are used when we cannot predict the impact or likelihood of the issue.

6 Issues

6.1 [Low] Potential use of stale WLD price in payout calculations

File(s): OperatorPayoutsWorldchain.sol

Description: The `transfer` and `transferWithMinimumWLDPrice` functions fetch the WLD token price via Chainlink's `latestRoundData`. However, they do not verify the freshness of this data using the `updatedAt` timestamp. As a result, the function may rely on stale prices, leading to inaccurate conversions during payout calculations.

```
function transfer(...) external onlyRelayer {
    if (to == address(0)) revert ZeroAddress();
    // @audit Potential stale price
    (, int256 answer,,,) = ORACLE.latestRoundData();
    if (answer <= 0) revert InvalidOraclePrice(answer);

    // Add 2*18 decimals to the dollar amount to account for 18 decimals from oracle price and 18 decimals from WLD
    uint256 tokenAmount = (dollarAmount * (10 ** 36)) / uint256(answer);
    // ...
}
```

Recommendation(s): Add a check for the `updatedAt` field to ensure the returned price is not older than a defined threshold.

Status: Fixed

Update from the client: Fixed in commit 10d8ef6ed7945394ef7e2786eb0f35fd750b94d8

6.2 [Info] High `minWldPrice` can lead to zero `tokenAmount`

File(s): OperatorPayoutsWorldchain.sol

Description: In the `transferWithMinimumWLDPrice` function, the WLD price used for conversion is selected as the maximum between the oracle price and the `minWldPrice`. However, if `minWldPrice` is set to an unreasonably high value, it can cause the computed `tokenAmount` to round down to zero. This would result in zero tokens being transferred, despite a nonzero `dollarAmount` input.

```
function transferWithMinimumWLDPrice(
    address payable to,
    uint256 dollarAmount,
    uint256 minWldPrice
) external onlyRelayer {
    if (to == address(0)) revert ZeroAddress();

    (, int256 answer,,,) = ORACLE.latestRoundData();
    if (answer <= 0) revert InvalidOraclePrice(answer);

    uint256 wldPrice = Math.max(uint256(answer), minWldPrice);

    // @audit tokenAmount may round to zero if minWldPrice is too high
    uint256 tokenAmount = dollarAmount * (10 ** 36) / wldPrice;
    // ...
}
```

Recommendation(s): Add a check to ensure that `tokenAmount` is non zero before proceeding with the transfer.

Status: Fixed

Update from the client: Fixed in commit 5183e109d99a01592ced87763d49db395ec64607

6.3 [Best Practices] Missing event emission in the `setRelayer` function

File(s): `OperatorPayoutsWorldchain.sol`

Description: The `setRelayer` function updates the relayer address but does not emit an event when this change occurs. Emitting events for important parameter updates is a best practice, as it allows off-chain monitoring and improves transparency.

```
function setRelayer(address _relayer) external onlyOwner {
    if (_relayer == address(0)) revert ZeroAddress();

    relayer = _relayer;
    //@audit Missing event emission
}
```

Recommendation(s): Emit an event in the `setRelayer` function to log changes to the relayer address.

Status: Fixed

Update from the client: Fixed in commit ff7b4bc6b9b68778329c5d3d1babdbbcdb8993dc

6.4 [Best Practices] Missing validation for zero `dollarAmount`

File(s): `OperatorPayoutsWorldchain.sol`

Description: Both `transfer` and `transferWithMinimumWLDPrice` functions lack a check to prevent execution when `dollarAmount` is zero. In such cases, the computed `tokenAmount` would also be zero, resulting in unnecessary execution and gas consumption.

Recommendation(s): Implement a check to ensure `dollarAmount` is not zero.

Status: Fixed

Update from the client: Fixed in commits 574a7ec499a4136998ddbb92e8ab1c783661db24 and 156e93bee5b08f9ca46ba80f06929ed4fa23c4c0

7 Documentation Evaluation

Software documentation refers to the written or visual information that describes the functionality, architecture, design, and implementation of software. It provides a comprehensive overview of the software system and helps users, developers, and stakeholders understand how the software works, how to use it, and how to maintain it. Software documentation can take different forms, such as user manuals, system manuals, technical specifications, requirements documents, design documents, and code comments. Software documentation is critical in software development, enabling effective communication between developers, testers, users, and other stakeholders. It helps to ensure that everyone involved in the development process has a shared understanding of the software system and its functionality. Moreover, software documentation can improve software maintenance by providing a clear and complete understanding of the software system, making it easier for developers to maintain, modify, and update the software over time. Smart contracts can use various types of software documentation. Some of the most common types include:

- Technical whitepaper: A technical whitepaper is a comprehensive document describing the smart contract's design and technical details. It includes information about the purpose of the contract, its architecture, its components, and how they interact with each other;
- User manual: A user manual is a document that provides information about how to use the smart contract. It includes step-by-step instructions on how to perform various tasks and explains the different features and functionalities of the contract;
- Code documentation: Code documentation is a document that provides details about the code of the smart contract. It includes information about the functions, variables, and classes used in the code, as well as explanations of how they work;
- API documentation: API documentation is a document that provides information about the API (Application Programming Interface) of the smart contract. It includes details about the methods, parameters, and responses that can be used to interact with the contract;
- Testing documentation: Testing documentation is a document that provides information about how the smart contract was tested. It includes details about the test cases that were used, the results of the tests, and any issues that were identified during testing;
- Audit documentation: Audit documentation includes reports, notes, and other materials related to the security audit of the smart contract. This type of documentation is critical in ensuring that the smart contract is secure and free from vulnerabilities.

These types of documentation are essential for smart contract development and maintenance. They help ensure that the contract is properly designed, implemented, and tested, and they provide a reference for developers who need to modify or maintain the contract in the future.

Remarks about the Worldchain Operator Payouts documentation

Worldcoin's smart contract documentation was provided within the PR description, which thoroughly explained the contract's purpose and implementation logic.

Each function in the codebase is accompanied by NatSpec comments that describe its purpose and input parameters. Additionally, the Worldcoin team provided a comprehensive walkthrough of the project during the kick-off call, addressing all questions and concerns raised by the Nethermind Security team.

8 Test Suite Evaluation

The Worldchain Operator Payouts's test suite is robust and clearly outlines the contract specifications. The test suite catches changes to the business logic. To assess the test coverage for these files, the Nethermind Security team applied a technique called mutation testing to uncover the untested paths. This technique introduces slight modifications to the code called "mutations" or "mutants." An example of a mutation is, for example, changing the operator in an expression `(a + b)` to `(a - b)`, or removing a `require(a > b)` statement entirely from the code. With these changes, the code no longer follows the expected business logic of the application, and the test suite should reflect that by failing.

Evaluation of the test suite with mutation testing consists of two phases:

a. Generating the modified version of each contract, called "mutants."

b. Inserting the mutant into the original codebase and running the test suite.

Only one modification can be tested at a time. If the contract has ten mutations, the test suite must run ten times (once for every mutation). If any of the tests fail, it means that the test suite caught the change in the code. Whenever that happens, the particular mutant is considered "slain" or "killed" and is removed from the mutant's set. If that does not occur, a new test case can be added to cover the code branch to "kill" the mutant.

The following table outlines the results of the analysis performed on the Worldchain Operator Payouts' core smart contract. The first column lists the contracts that were tested using mutation testing. The second column indicates the number of mutants generated and how many were "slain" (i.e., caught by the test suite). The third column provides the percentage of mutants slain, reflecting the effectiveness of the test suite in covering the particular contract. The higher the score, the better the test suite is at finding bugs.

Contract	Mutants (slain / total generated)	Score
`OperatorPayoutsWorldchain.sol`	72 / 74	97.29%
Total	72 / 74	97.29%

The following is a list of code modifications not caught by the existing test suite. Each point highlights a scenario that could benefit from higher test coverage to enhance the protocol's overall security and resilience in the next code change iterations as the project evolves:

OperatorPayoutsWorldchain.sol

- The `transferWithMinimumWLDPrice(...)` function is only tested in scenarios where the provided `to` address is non-zero. Consider adding a test case that triggers the `ZeroAddress` error.
- The `transferWithMinimumWLDPrice(...)` function doesn't test the scenarios where the `answer` value returned by the oracle is zero or less. Consider adding a test case that triggers the `InvalidOraclePrice` error.

Remarks about the Worldchain Operator Payouts Test Suite

The test suite for the Worldchain Operator Payouts Protocol is robust in terms of unit testing for core functionality. Changes in business logic are effectively caught, ensuring that key components behave as expected under various conditions. However, the suite currently relies on mock contracts, such as the mock implementation of the allowance module. While this approach is useful for isolating components during unit testing, it limits the ability to detect integration issues that may arise in production. The test suite can be improved by incorporating more end-to-end tests using actual contract implementations and by addressing the missing flows outlined earlier.

8.1 Automated Tools

8.1.1 AuditAgent

All the relevant issues raised by the AuditAgent have been incorporated into this report. The AuditAgent is an AI-powered smart contract auditing tool that analyses code, detects vulnerabilities, and provides actionable fixes. It accelerates the security analysis process, complementing human expertise with advanced AI models to deliver efficient and comprehensive smart contract audits. Available at https://app.auditagent.nethermind.io.

9 About Nethermind

Nethermind is a Blockchain Research and Software Engineering company. Our work touches every part of the web3 ecosystem - from layer 1 and layer 2 engineering, cryptography research, and security to application-layer protocol development. We offer strategic support to our institutional and enterprise partners across the blockchain, digital assets, and DeFi sectors, guiding them through all stages of the research and development process, from initial concepts to successful implementation.

We offer security audits of projects built on EVM-compatible chains and Starknet. We are active builders of the Starknet ecosystem, delivering a node implementation, a block explorer, a Solidity-to-Cairo transpiler, and formal verification tooling. Nethermind also provides strategic support to our institutional and enterprise partners in blockchain, digital assets, and decentralized finance (DeFi). In the next paragraphs, we introduce the company in more detail.

Blockchain Security: At Nethermind, we believe security is vital to the health and longevity of the entire Web3 ecosystem. We provide security services related to Smart Contract Audits, Formal Verification, and Real-Time Monitoring. Our Security Team comprises blockchain security experts in each field, often collaborating to produce comprehensive and robust security solutions. The team has a strong academic background, can apply state-of-the-art techniques, and is experienced in analyzing cutting-edge Solidity and Cairo smart contracts, such as ArgentX and StarkGate (the bridge connecting Ethereum and StarkNet). Most team members hold a Ph.D. degree and actively participate in the research community, accounting for 240+ articles published and 1,450+ citations in Google Scholar. The security team adopts customer-oriented and interactive processes where clients are involved in all stages of the work.

Blockchain Core Development: Our core engineering team, consisting of over 20 developers, maintains, improves, and upgrades our flagship product - the Nethermind Ethereum Execution Client. The client has been successfully operating for several years, supporting both the Ethereum Mainnet and its testnets, and now accounts for nearly a quarter of all synced Mainnet nodes. Our unwavering commitment to Ethereum's growth and stability extends to sidechains and layer 2 solutions. Notably, we were the sole execution layer client to facilitate Gnosis Chain's Merge, transitioning from Aura to Proof of Stake (PoS), and we are actively developing a full-node client to bolster Starknet's decentralization efforts. Our core team equips partners with tools for seamless node set-up, using generated docker-compose scripts tailored to their chosen execution client and preferred configurations for various network types.

DevOps and Infrastructure Management: Our infrastructure team ensures our partners' systems operate securely, reliably, and efficiently. We provide infrastructure design, deployment, monitoring, maintenance, and troubleshooting support, allowing you to focus on your core business operations. Boasting extensive expertise in Blockchain as a Service, private blockchain implementations, and node management, our infrastructure and DevOps engineers are proficient with major cloud solution providers and can host applications in-house or on clients' premises. Our global in-house SRE teams offer 24/7 monitoring and alerts for both infrastructure and application levels. We manage over 5,000 public and private validators and maintain nodes on major public blockchains such as Polygon, Gnosis, Solana, Cosmos, Near, Avalanche, Polkadot, Aptos, and StarkWare L2. Sedge is an open-source tool developed by our infrastructure experts, designed to simplify the complex process of setting up a proof-of-stake (PoS) network or chain validator. Sedge generates docker-compose scripts for the entire validator set-up based on the chosen client, making the process easier and quicker while following best practices to avoid downtime and being slashed.

Cryptography Research: At Nethermind, our Cryptography Research team is dedicated to continuous internal research while fostering close collaboration with external partners. The team has expertise across a wide range of domains, including cryptography protocols, consensus design, decentralized identity, verifiable credentials, Sybil resistance, oracles, and credentials, distributed validator technology (DVT), and Zero-knowledge proofs. This diverse skill set, combined with strong collaboration between our engineering teams, enables us to deliver cutting-edge solutions to our partners and clients.

Smart Contract Development & DeFi Research: Our smart contract development and DeFi research team comprises 40+ world-class engineers who collaborate closely with partners to identify needs and work on value-adding projects. The team specializes in Solidity and Cairo development, architecture design, and DeFi solutions, including DEXs, AMMs, structured products, derivatives, and money market protocols, as well as ERC20, 721, and 1155 token design. Our research and data analytics focuses on three key areas: technical due diligence, market research, and DeFi research. Utilizing a data-driven approach, we offer in-depth insights and outlooks on various industry themes.

Our suite of L2 tooling: Warp is Starknet's approach to EVM compatibility. It allows developers to take their Solidity smart contracts and transpile them to Cairo, Starknet's smart contract language. In the short time since its inception, the project has accomplished many achievements, including successfully transpiling Uniswap v3 onto Starknet using Warp.

- **Voyager** is a user-friendly Starknet block explorer that offers comprehensive insights into the Starknet network. With its intuitive interface and powerful features, Voyager allows users to easily search for and examine transactions, addresses, and contract details. As an essential tool for navigating the Starknet ecosystem, Voyager is the go-to solution for users seeking in-depth information and analysis;
- **Horus** is an open-source formal verification tool for StarkNet smart contracts. It simplifies the process of formally verifying Starknet smart contracts, allowing developers to express various assertions about the behavior of their code using a simple assertion language;
- **Juno** is a full-node client implementation for Starknet, drawing on the expertise gained from developing the Nethermind Client. Written in Golang and open-sourced from the outset, Juno verifies the validity of the data received from Starknet by comparing it to proofs retrieved from Ethereum, thus maintaining the integrity and security of the entire ecosystem.

Learn more about us at nethermind.io.

General Advisory to Clients

As auditors, we recommend that any changes or updates made to the audited codebase undergo a re-audit or security review to address potential vulnerabilities or risks introduced by the modifications. By conducting a re-audit or security review of the modified codebase, you can significantly enhance the overall security of your system and reduce the likelihood of exploitation. However, we do not possess the authority or right to impose obligations or restrictions on our clients regarding codebase updates, modifications, or subsequent audits. Accordingly, the decision to seek a re-audit or security review lies solely with you.

Disclaimer

This report is based on the scope of materials and documentation provided by you to Nethermind in order that Nethermind could conduct the security review outlined in **1. Executive Summary** and **2. Audited Files**. The results set out in this report may not be complete nor inclusive of all vulnerabilities. Nethermind has provided the review and this report on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Blockchain technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. This report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on this report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, Nethermind disclaims any liability in connection with this report, its content, and any related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Nethermind does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and Nethermind will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. 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. 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.