Quantstamp Security Assessment Certificate

October 7th 2022 — Quantstamp Verified

Catheon Token

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

Executive Summary

Type	ERC20 Token
Auditors	Martin Derka, Senior Research Engineer Zeeshan Meghji, Auditing Engineer Jeffrey Kam, Audit Engineer
Timeline	2022-10-04 through 2022-10-07
EVM	Paris
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Documentation Quality	Medium
Test Quality	Low

Source Code

Repository	Commit
CatheonGaming/cgc-evm-public-contract	6d5dfba initial audit

Goals

- Assessing the security of the token

Total Issues	**13** (1 Resolved)
High Risk Issues	0 (0 Resolved)
Medium Risk Issues	**1** (0 Resolved)
Low Risk Issues	**3** (0 Resolved)
Informational Risk Issues	**7** (1 Resolved)
Undetermined Risk Issues	**2** (0 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	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.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
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).
Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

The audited code is an implementation of an upgradeable ERC20 token. The implementation is rather standard, with fees being deducted upon transfers. Quantstamp discovered issues that relate to an ongoing operation of the contract and as such, the developers and community should be aware of. Quantstamp did not discover defects of high severity that could lead to the loss of users' tokens without additional external factors, such as the owner's key getting compromised. The reviewed contract appears to be well tested as an implementation, but as the token is expected to be deployed as a proxy, the tests should include testing the proxy interaction as well. This is currently not the case, and the auditors urge for adding such a test suite (this is reflected in the overall test quality assessment as low) . The provided documentation is deemed rather reasonable, but contains a few nuances that would deserve to be mentioned explicitly.

ID	Description	Severity	Status
QSP-1	Privileged Roles and Ownership	^ Medium	Unresolved
QSP-2	Ownership Can Be Renounced	⌄ Low	Unresolved
QSP-3	Missing Input Validation	⌄ Low	Unresolved
QSP-4	Configuration of Services	⌄ Low	Unresolved
QSP-5	Allowance Double-Spend Exploit	○ Informational	Mitigated
QSP-6	Users Can Subvert Fees for Small Transfers	○ Informational	Unresolved
QSP-7	Use of Non-Standard Decimal for the Token	○ Informational	Unresolved
QSP-8	Attacker Can Take Control of the Implementation Contract	○ Informational	Unresolved
QSP-9	Upgradeability	○ Informational	Unresolved
QSP-10	Event Parameters Not Indexed	○ Informational	Unresolved
QSP-11	Using `uint16` for the Fee Parameter	○ Informational	Unresolved
QSP-12	Redundant Pause Mechanism	? Undetermined	Unresolved
QSP-13	Burning Tokens Does Not Reduce Maximum Supply	? Undetermined	Unresolved

Quantstamp Audit Breakdown

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

DISCLAIMER:
The scope of this audit is restricted to the `catheon_token/contracts/CatheonToken.sol` contract only. If the final commit hash provided by the client contains features that are not within the scope of the audit or an associated fix review, those features 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

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. 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:
   i. 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.
   ii. 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, clarify, 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.

Toolset

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

Setup

Tool Setup:

- Slither v0.8.3

Steps taken to run the tools:

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

Findings

QSP-1 Privileged Roles and Ownership

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: The `CatheonToken` contract features an owner role with a high degree of control over the contract. Token holders must completely trust the contract owner not to behave maliciously and implement security best practices for the privileged account. We have listed all permissions of the contract owner below:

- Mint an arbitrary amount of tokens to any address at any time.
- Burn an arbitrary amount of tokens of any user at any time.
- Reset the treasury address, which receives all transfer fees.
- Set any account as a service, allowing the account to transfer tokens without paying fees.
- Change the token transfer fees to up to 90%.
- Reset the maximum supply of the token. Users have no guarantee that the token will not be hyperinflated.
- Pause the ability to mint new tokens.

Recommendation:

1. Consider removing any unnecessary privileged permissions of the contract owner. In particular, the ability to mint and burn tokens at any time should be removed if it is not completely necessary. We also recommend removing the ability to reset the maximum supply.
2. Privileged roles should be secured by large multi-signature wallets and assigned to time-lock contracts to allow users to opt-out of significant changes, such as increased token transfer fees.

This centralization of power needs to be made clear to the users, especially depending on the level of privilege the contract allows to the owner.

QSP-2 Ownership Can Be Renounced

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed.

Recommendation: Double check if this is the intended behaviour. If this is an issue, override the function to disable the ownership renounce functionality. If not, consider including an explanation in the documentation and code.

QSP-3 Missing Input Validation

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: The `CatheonToken.initialize()` functions lacks several important validation checks upon the input parameters. This could result in the contract entering an unexpected state and not functioning as expected. We have listed all the missings checks below:

- `name_` and `symbol_` should not be empty strings.
- `initialBalance_` should be less than the default maximum supply, currently `10**19`.
- `treasury_` should not be the zero address.

Recommendation: Implement the suggested validation checks.

QSP-4 Configuration of Services

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: The token sends a fee to treasury upon transfers from and to accounts that are not configured as services. This can result in difficulties while integrating the token with services such

as decentralized exchanges, lending pools, and other protocols. The operator of the token needs to register all the addresses that should not lose tokens on transfers as services, otherwise token accounting inside of third party protocols can fail. The configuration of service accounts require transacting, and as such, it is the subject of transaction ordering dependence, which should be communicated to the users as well.

Recommendation: Document the fee on transfer properly for all the third parties who use the Catheon token. Document the transaction ordering dependence of configuring services for the users. Ensure that you register the proper addresses as services so that they do not pay fees on transfer.

QSP-5 Allowance Double-Spend Exploit

Severity: *Informational*

Status: Mitigated

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario: 1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)

1. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
2. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
3. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
4. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`. Furthermore, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value.

QSP-6 Users Can Subvert Fees for Small Transfers

Severity: *Informational*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: There is no guarantee that `(amount * _feePercent) >= PERCENTAGE_DIVISION` in the `_transfer()` function. For example, suppose an owner wants to set the fee percentage to be 0.1% and calls setFee(1). If a user then calls transfer with an amount of 500, the calculated `feeAmount` on line 87 will be evaluated to 0. This means a user can essentially avoid the transfer fee by choosing a small enough amount to transfer based on the fee percentage.

Exploit Scenario:

```
await catheonToken.connect(owner).mint(alice.address, 500);
await catheonToken.connect(owner).mint(bob.address, 0);

// Setting transfer fee to 0.1%
const transfer_fee = 1;
await catheonToken.setFee(transfer_fee);

await expect(catheonToken.connect(alice).transfer(bob.address, 500))
  .to.emit(catheonToken, "Transfer")
  .withArgs(alice.address, treasury.address, 0);

expect(await catheonToken.balanceOf(treasury.address)).to.eq(0);
expect(await catheonToken.balanceOf(alice.address)).to.eq(0);
expect(await catheonToken.balanceOf(bob.address)).to.eq(500);
```

Alice and Bob are not service addresses, but the transfer amount of 500 from Alice to Bob does not incur the expected transferral fee.

Recommendation: We classify the issue as informational, because the fee lost in a single transfer is limited to `900 * 10^-9` (the maximum fee accounting for the decimals). The gas cost of such a transfer would likely outweigh the benefit, and thus the attack would not be profitable. A potential mitigation is to only allow users to transfer amounts divisible by `PERCENTAGE_DIVISION` to ensure such fee avoidance, or more generally the underlying rounding issue, is not possible.

QSP-7 Use of Non-Standard Decimal for the Token

Severity: *Informational*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: The token uses 9 decimals instead of the default and customary 18 decimals, but this is not mentioned in the documentation.

Recommendation: Make sure the users are aware of this by including it in the documentation.

QSP-8 Attacker Can Take Control of the Implementation Contract

Severity: *Informational*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: Upgradeability often involves two different kinds of contracts: implementation contracts and proxies. The `CatheonToken` contract is an implementation contract, and it is currently not automatically initialized. If an implementation contract is not initialized, a malicious user could call the `initialize()` function on the implementation contract to take ownership of it. If the implementation contract uses delegate calls, it may be possible for an attacker to self-destruct the contract.
The auditors did not find a way how controlling the implementation of the token could result in users loosing funds. Therefore, the issue is classified as informational. Its severity could be higher otherwise.

Recommendation: Add a constructor to the `CatheonToken` contract, which calls `_disableInitializers()` to automatically initialize the contract and block a malicious user from taking control of it.

QSP-9 Upgradeability

Severity: *Informational*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: The `CatheonToken` contract is an upgradeable implementation contract. While this is not a vulnerability, users should be aware that the behaviour of the token could drastically change if the contract is upgraded. Furthermore, new vulnerabilities not present during the audit could be introduced in upgraded versions of the contract.

Recommendation: The fact that the contract can be upgraded and reasons for future upgrades should be communicated to users beforehand.

QSP-10 Event Parameters Not Indexed

Severity: *Informational*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: All events defined by the `CatheonToken` contract do not have any parameters indexed. Indexing parameters of events allows for easier querying of event data and can thus improve monitoring for the application. Monitoring contracts is key to contract security and can help developers detect exploits earlier. We have listed all the events which should have their key parameters indexed below:

- `SetPaused`
- `SetTreasury`
- `SetService`
- `SetFeePercent`
- `SetMaxSupply`

Recommendation: Index the key parameters of all events.

QSP-11 Using `uint16` for the Fee Parameter

Severity: *Informational*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: The smart contract records the fee percentage as `uint16`. The fee is recorded in the smart contract storage as a regular state attribute, not inside of a struct. Hence, the value will be stored as a full word with 256 bits, resulting in no gas savings. On the other hand, when performing computations that involve a fee, the type may result in an additional overhead cost due to implicit conversion to `uint256`.

Recommendation: The auditors recommend storing the fee value as `uint256`.

QSP-12 Redundant Pause Mechanism

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: The `CatheonToken` contract features a pause mechanism that allows the contractor to pause the minting of tokens. Pausing and unpausing are done when the contract owner calls the `setPaused()` function. However, only the contract owner can mint tokens. It is redundant to have a pause mechanism that is controlled by the same account that is also responsible for minting the tokens. A more meaningful pause mechanism would assign the permission for pausing and unpausing the contract to a different role than the one which also mints the tokens. The current pausing mechanism has no effect on token transfers.

Recommendation: Either remove the ineffective pause mechanism or assign permission for the `setPaused()` function to an account that is not the contract's owner. Confirm that pausing should have no effect on the transfers of the tokens.

QSP-13 Burning Tokens Does Not Reduce Maximum Supply

Severity: *Undetermined*

Status: Unresolved

File(s) affected: `catheon_token/contracts/CatheonToken.sol`

Description: A common use case of burning tokens is to reduce the maximum supply of the token. When the `CatheonToken.burn()` function is called, the token's total supply is reduced, but the maximum supply remains the same.

Recommendation: If not reducing the maximum supply is the desired outcome of burning tokens, then no changes are required. Otherwise, the maximum supply should be decreased by the number of tokens burned.

Automated Analyses

Slither

Slither did not report any issues that the auditors would not consider benign, or that would not be already included in this report.

Adherence to Specification

The code adheres to the provided specification.

Code Documentation

1. It is not clear from the documentation whether a user should be allowed to transfer directly to the treasury. Furthermore, it is unclear whether such user-to-treasury (or treasury-to-user) transactions should be subjected to the transferral fee since it is not mentioned whether the treasury address is considered a service address. In the current implementation, the transfer fee is not applied for such user-to-treasury (or treasury-to-user) transactions. Consider documenting the intended logic.
2. It is unclear what `s_mintingFinished` is referring to in the comment on `CatheonToken.sol#L160`.
3. Add code comments to elucidate the keys and values of mappings. For example, this comment could be used to document the `services` mapping: `// (address => isService)`.
4. All events should be documented using the NatSpec standard.

Adherence to Best Practices

The auditors highlight the following best practice violations:

1. `CatheonToken.sol`: The private variable paused on line 20 is inconsistent with other private variables naming.
2. `CatheonToken.sol`: Inconsistent capitalization in comments. Specifically, lines 55, 64, 69, 132, 137, 142, and 156 should be capitalized after `@dev`.
3. `CatheonToken.sol`: The function descriptions on line 132, 137, and 142 should include the return descriptions.
4. `CatheonToken.sol`: The variable naming for paused on line 20 can be more descriptive to indicate this is only for pausing the minting functionality, and not to be confused with other pausing mechanisms, such as for pausing token transfer in the `ERC20Pausable` contract.
5. `CatheonToken.sol`: Line 250 of `test/index.ts` should be "8. Burn Token" and similarly line 251 should be "8.1 Only the owner can burn token".
6. `CatheonToken.sol`: Consider renaming the storage variable `paused` to `pausedMinting` for clarity.
7. `CatheonToken.sol`: Replace magic constants with named constants such as in the case of `0` and `900` in `CatheonToken.sol#L134:require(percentage != 0 && percentage <= 900, "Invalid Fee Percentage");`.

Test Results

Test Suite Results

All the tests included with the codebase pass. The tests do not include assertions for the emitted events, but they confirm the rest of the intended logic. Additionally, the tests only test the implementation. The interaction via a proxy is entirely untested.

```
$ hardhat coverage --solcoverjs ./.solcover.js --temp build --network hardhat

Version
=======
> solidity-coverage: v0.8.1

Instrumenting for coverage...
=============================

> CatheonToken.sol

Compilation:
============

Generating typings for: 8 artifacts in dir: ./build/typechain for target: ethers-v5
Successfully generated 38 typings!
Compiled 8 Solidity files successfully

Network Info
============
> HardhatEVM: v2.11.1
> network:    hardhat



  Catheon Token Unit Tests
    ✓ 1. Token can not deploy with zero-initial_supply or zero-treasury_address (102ms)
    ✓ 2. After deploying, token should be have correct states (62ms)
  3. Fee percentage
    ✓ 3.1 The fee percentage can be set by only owner (49ms)
    ✓ 3.2 The fee percentage can not be set as origin value
    ✓ 3.3 The fee percentage can not be set as 0% or over 90%
  4. Set service address
    ✓ 4.1 The service address can be set by only owner (42ms)
    ✓ 4.2 The service address can not be set as origin value
  5. Set treasury
    ✓ 5.1 The treasury can be set by only owner
    ✓ 5.2 The service address can not be set as origin value or zero address
  6. Minting Token
    ✓ 6.1 Only the owner can mint token
    ✓ 6.2 Only the owner can disable minting (74ms)
    ✓ 6.3 Only the owner can set max supply
    ✓ 6.4 The owner can not set max_supply with the smaller value than current total_supply
    ✓ 6.5 The owner can not mint token over max supply
  7. Transfer
    ✓ 7.1 if token is transferring from/to serviceAddress, The fee is not be applied (40ms)
    ✓ 7.2 if token-transfer is not related with services, The fee should be applied
  6. Burn Token
    ✓ 6.1 Only the owner can burn token (48ms)


17 passing (4s)
```

Code Coverage

The implementation is sufficiently covered with tests. The interaction via a proxy is entirely untested.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts/	100	97.37	100	100	
CatheonToken.sol	100	97.37	100	100	
All files	**100**	**97.37**	**100**	**100**	

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.

Contracts

```
e0b2196a725324ed6233d11d63ba186e864459a075a616a5b7131fcf759d6ee1 ./contracts/CatheonToken.sol
```

Tests

```
493cb24c75f28b4e8b5fbee15a4143ca5a44aae33cff1eced6a1e9cc6e4a5feb ./test/helper.ts
00879cd2d7a485ea8357b55466c8016d0d352c7327f2702b80a82ee1edd7e0ef ./test/index.ts
```

Changelog

- 2022-10-07 - Initial report

About Quantstamp

Quantstamp is a global leader in blockchain security backed by Pantera, Softbank, and Commonwealth among other preeminent investors. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its white glove security and risk assessment services.

The team consists of web3 thought leaders hailing from top organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Many of the auditors 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 250 audits and secured over $200 billion in digital asset risk from hackers. In addition to providing an array of security services, Quantstamp facilitates the adoption of blockchain technology through strategic investments within the ecosystem and acting as a trusted advisor to help projects scale.

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, Aave, 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.

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, Inc. (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 the website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided 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. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports 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, we disclaim all warranties, expressed or 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. We do 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 we 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.