



Security Assessment



Aqua Pepe Token

June 25, 2023





Audit Status: Pass

Audit Edition: Advance






























Risk Analysis

Classifications of Manual Risk Results

Classification	Description
 Critical	Danger or Potential Problems.
 Major	Be Careful or Fail test.
 Minor	Pass, Not-Detected or Safe Item.
 Informational	Function Detected

Manual Code Review Risk Results

Contract Privilege	Description
 Buy Tax	5
 Sale Tax	5
 Cannot Sale	Pass
 Cannot Sale	Pass
 Max Tax	5
 Modify Tax	Not Detected
 Fee Check	Pass
 Is Honeypot?	Not detected
 Trading Cooldown	Not Detected
 Can Pause Trade?	Fail
 Pause Transfer?	Detected, owner needs to enable trade

Contract Priviledge	Description
 Max Tx?	Pass
 Is Anti Whale?	Not Detected
 Is Anti Bot?	Not Detected
 Is Blacklist?	Not Detected
 Blacklist Check	Pass
 is Whitelist?	Detected
 Can Mint?	Pass
 Is Proxy?	Not Detected
 Can Take Ownership?	Not detected
 Hidden Owner?	Not detected
 Owner	0x7301B8436F398fddBa7d1E1CF0982186A47A0822
 Self Destruct?	Not Detected
 Other?	Not detected
 Other?	Not detected
 Holders	1
 Auditor Confidence	High

The following quick summary it's added to the project overview; however, there are more details about the audit and its results. Please read every detail.

Project Overview

Token Summary

Parameter	Result
Address	0x8696EA4C470D854e3F841a18Fd9E40d2Eddf4C61
Name	Aqua Pepe
Token Tracker	Aqua Pepe (APEPE)
Decimals	18
Supply	1,000,000,000,000
Platform	Binance Smart Chain
compiler	v0.8.9+commit.e5eed63a
Contract Name	AquaPepe
Optimization	Yes with 200 runs
LicenseType	MIT
Language	Solidity

Main Contract Assessed

Contract Name

Name	Contract	Live
Aqua Pepe	0x8696EA4C470D854e3F841a18Fd9E40d2Eddf4C61	Yes

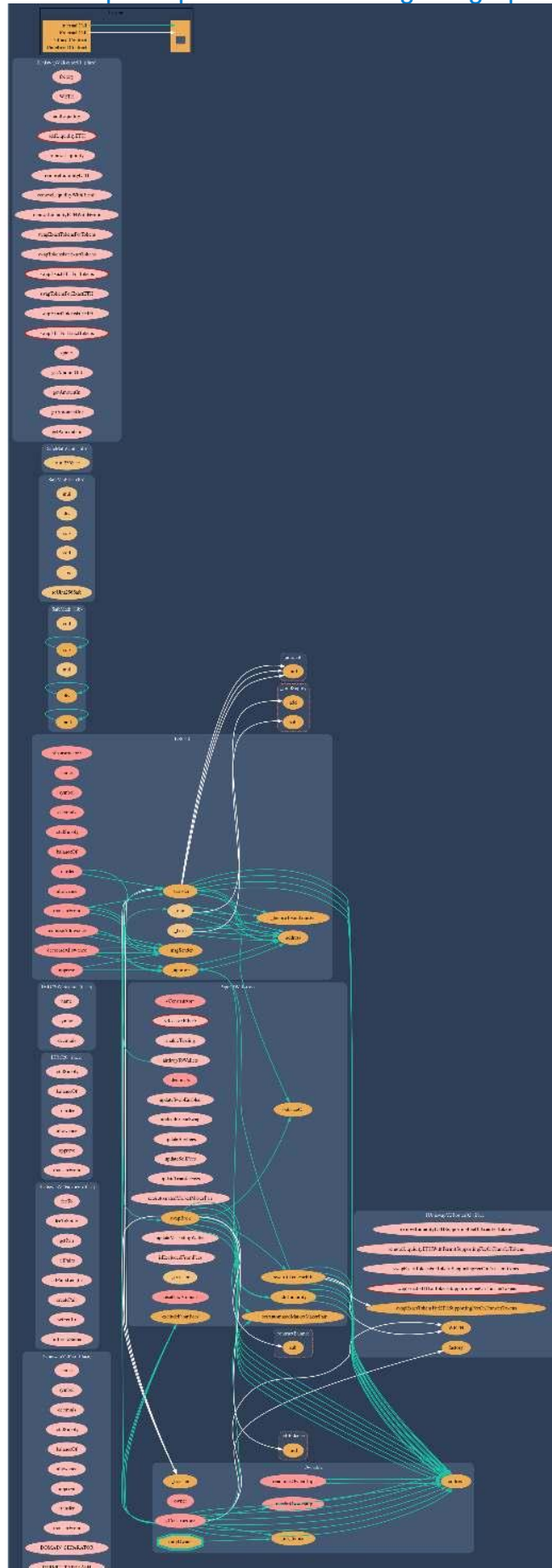
TestNet Contract Assessed

Contract Name

Name	Contract	Live
Aqua Pepe	0x01b79db472b4ba5cf930cdb7aa34052d5e73989a	Yes

Call Graph

The contract for Aqua Pepe has the following call graph structure.



Smart contract Vulnerability Checks

The Smart Contract Weakness Classification Registry (SWC Registry) is an implementation of the weakness classification scheme proposed in EIP-1470. It is loosely aligned to the terminologies and structure used in the Common Weakness Enumeration (CWE) while overlaying a wide range of weakness variants that are specific to smart contracts.

ID	Severity	Name	File	location
SWC-100	Pass	Function Default Visibility	Aqua Pepe	L: 0 C: 0
SWC-101	Pass	Integer Overflow and Underflow.	Aqua Pepe	L: 0 C: 0
SWC-102	Pass	Outdated Compiler Version file.	Aqua Pepe	L: 0 C: 0
SWC-103	Low	A floating pragma is set.	Aqua Pepe	L: 7 C: 0
SWC-104	Pass	Unchecked Call Return Value.	Aqua Pepe	L: 0 C: 0
SWC-105	Pass	Unprotected Ether Withdrawal.	Aqua Pepe	L: 0 C: 0
SWC-106	Pass	Unprotected SELFDESTRUCT Instruction	Aqua Pepe	L: 0 C: 0
SWC-107	Pass	Read of persistent state following external call.	Aqua Pepe	L: 0 C: 0
SWC-108	Pass	State variable visibility is not set..	Aqua Pepe	L: 0 C: 0
SWC-109	Pass	Uninitialized Storage Pointer.	Aqua Pepe	L: 0 C: 0
SWC-110	Pass	Assert Violation.	Aqua Pepe	L: 0 C: 0

BLANK

ID	Severity	Name	File	location
SWC-111	Pass	Use of Deprecated Solidity Functions.	Aqua Pepe	L: 0 C: 0
SWC-112	Pass	Delegate Call to Untrusted Callee.	Aqua Pepe	L: 0 C: 0
SWC-113	Pass	Multiple calls are executed in the same transaction.	Aqua Pepe	L: 0 C: 0
SWC-114	Pass	Transaction Order Dependence.	Aqua Pepe	L: 0 C: 0
SWC-115	Pass	Authorization through tx.origin.	Aqua Pepe	L: 0 C: 0
SWC-116	Pass	A control flow decision is made based on The block.timestamp environment variable.	Aqua Pepe	L: 0 C: 0
SWC-117	Pass	Signature Malleability.	Aqua Pepe	L: 0 C: 0
SWC-118	Pass	Incorrect Constructor Name.	Aqua Pepe	L: 0 C: 0
SWC-119	Pass	Shadowing State Variables.	Aqua Pepe	L: 0 C: 0
SWC-120	Pass	Potential use of block.number as source of randomness.	Aqua Pepe	L: 0 C: 0
SWC-121	Pass	Missing Protection against Signature Replay Attacks.	Aqua Pepe	L: 0 C: 0
SWC-122	Pass	Lack of Proper Signature Verification.	Aqua Pepe	L: 0 C: 0
SWC-123	Pass	Requirement Violation.	Aqua Pepe	L: 0 C: 0
SWC-124	Pass	Write to Arbitrary Storage Location.	Aqua Pepe	L: 0 C: 0

ID	Severity	Name	File	location
SWC-125	Pass	Incorrect Inheritance Order.	Aqua Pepe	L: 0 C: 0
SWC-126	Pass	Insufficient Gas Griefing.	Aqua Pepe	L: 0 C: 0
SWC-127	Pass	Arbitrary Jump with Function Type Variable.	Aqua Pepe	L: 0 C: 0
SWC-128	Pass	DoS With Block Gas Limit.	Aqua Pepe	L: 0 C: 0
SWC-129	Pass	Typographical Error.	Aqua Pepe	L: 0 C: 0
SWC-130	Pass	Right-To-Left-Override control character (U +202E).	Aqua Pepe	L: 0 C: 0
SWC-131	Pass	Presence of unused variables.	Aqua Pepe	L: 0 C: 0
SWC-132	Pass	Unexpected Ether balance.	Aqua Pepe	L: 0 C: 0
SWC-133	Pass	Hash Collisions with Multiple Variable Length Arguments.	Aqua Pepe	L: 0 C: 0
SWC-134	Pass	Message call with hardcoded gas amount.	Aqua Pepe	L: 0 C: 0
SWC-135	Pass	Code With No Effects (Irrelevant/Dead Code).	Aqua Pepe	L: 0 C: 0
SWC-136	Pass	Unencrypted Private Data On-Chain.	Aqua Pepe	L: 0 C: 0

We scan the contract for additional security issues using MYTHX and industry-standard security scanning tools.

Smart Contract VulnerabilityDetails

SWC-103 - Floating Pragma.

CWE-664: Improper Control of a Resource Through its Lifetime.

References:

Description:

Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.

Remediation:

Lock the pragma version and also consider known bugs (<https://github.com/ethereum/solidity/releases>) for the compiler version that is chosen.

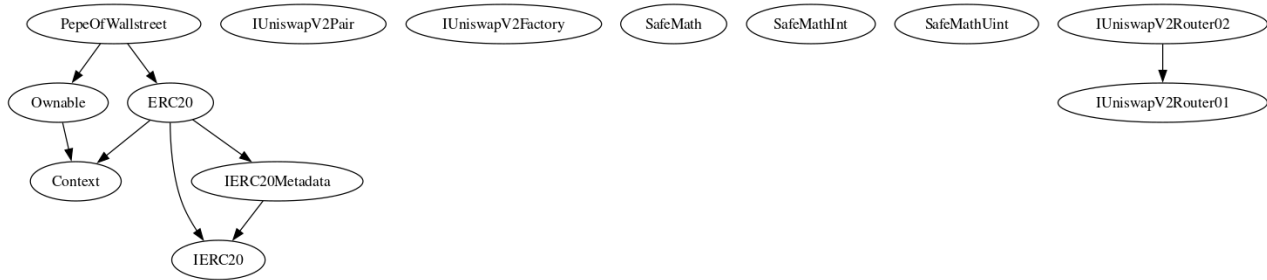
Pragma statements can be allowed to float when a contract is intended for consumption by other developers, as in the case with contracts in a library or EthPM package. Otherwise, the developer would need to manually update the pragma in order to compile locally.

References:

Ethereum Smart Contract Best Practices - Lock pragmas to specific compiler version.

Inheritance

The contract for Aqua Pepe has the following inheritance structure.





Smart Contract Advance Check

ID	Severity	Name	Result	Status
PWS-01	Minor	Potential Sandwich Attacks.	Pass	Not-Found
PWS-02	Minor	Function Visibility Optimization	Pass	Detected
PWS-03	Minor	Lack of Input Validation.	Pass	Detected
PWS-04	Major	Centralized Risk In addLiquidity.	Pass	Detected
PWS-05	Minor	Missing Event Emission.	Pass	Detected
PWS-06	Minor	Conformance with Solidity Naming Conventions.	Pass	Detected
PWS-07	Minor	State Variables could be Declared Constant.	Pass	Not-Found
PWS-08	Minor	Dead Code Elimination.	Pass	Not-Found
PWS-09	Major	Third Party Dependencies.	Pass	Not-Found
PWS-10	Major	Initial Token Distribution.	Pass	Not-Found
PWS-11	Minor	Multisend is present in code.	Pass	Detected
PWS-12	Major	Centralization Risks In The X Role	Pass	Not-Found
PWS-13	Informational	Extra Gas Cost For User..	Pass	Not-Found
PWS-14	Medium	Unnecessary Use Of SafeMath	Fail	Detected
PWS-15	Medium	Symbol Length Limitation due to Solidity Naming Standards.	Pass	Not-Found

ID	Severity	Name	Result	Status
PWS-16	Medium	Invalid collection of Taxes during Transfer.	Pass	Not-Found
PWS-17	Informational	Conformance to numeric notation best practice.	Pass	Not-Found
PWS-18	Critical	Stop Transactions by using Enable Trade.	Pass	Detected

PWS-14 Unnecessary Use Of SafeMath

Category	Severity	Location	Status
Logical Issue	 Medium	Aqua Pepe.sol: 7,9	 Detected

Description

The SafeMath library is used unnecessarily. With Solidity compiler versions 0.8.0 or newer, arithmetic operations will automatically revert in case of integer overflow or underflow.

```
library SafeMath {  
    An implementation of SafeMath library is found.  
    using SafeMath for uint256;  
    SafeMath library is used for uint256 type in contract.
```






Remediation

We advise removing the usage of SafeMath library and using the built-in arithmetic operations provided by the Solidity programming language






Project Action - Nil

Technical Findings Summary

Classification of Risk

Severity	Description
 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.
 Major	Risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
 Medium	Risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform
 Minor	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.
 Informational	Errors are often recommended to improve the code's style or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

Findings

Severity	Found	Pending	Resolved
 Critical	0	0	0
 Major	1	0	0
 Medium	0	0	0
 Minor	0	0	0
 Informational	0	0	0
Total	1	0	0

Social Media Checks

Social Media	URL	Result
Twitter	https://twitter.com/aquapepe	Pass
Other		Fail
Website		Pass
Telegram	https://t.me/aquapepe	Pass

We recommend to have 3 or more social media sources including a completed working websites.

Social Media Information Notes:

Auditor Notes: undefined

Project Owner Notes:



Assessment Results

Score Results

Review	Score
Overall Score	81/100
Auditor Score	87/100
Review by Section	Score
Manual Scan Score	33/53
SWC Scan Score	36 /37
Advance Check Score	12 /19

The Following Score System Has been Added to this page to help understand the value of the audit, the maximum score is 100, however to attain that value the project must pass and provide all the data needed for the assessment. Our Passing Score has been changed to 80 Points, if a project does not attain 80% is an automatic failure. Read our notes and final assessment below.

Audit Passed



Assessment Results

Important Notes:

- Contract has taxes up to 10%.
- Owner can't set max tx amount.
- No high-risk Exploits/Vulnerabilities Were Found in the Source Code.
- Contract has been developed by Brick and follow the coding best practices, we have fully tested the code and its functionalities.

Auditor Score =87
Audit Passed



Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

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

Volatile Code

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

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different requirements on the input variables than a setter function.

Coding Best Practices

ERC 20 Coding Standards are a set of rules that each developer should follow to ensure the code meet a set of criteria and is readable by all the developers.



Disclaimer

ETHER AUTHORITY has conducted an independent security assessment to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the reviewed code for the scope of this assessment. This report does not constitute agreement, acceptance, or advocacy for the Project, and users relying on this report should not consider this as having any merit for financial advice in any shape, form, or nature. The contracts audited do not account for any economic developments that the Project in question may pursue, and the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude, and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are entirely free of exploits, bugs, vulnerabilities or deprecation of technologies.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence, regardless of the findings presented. Information is provided 'as is, and ETHER AUTHORITY is under no covenant to audited completeness, accuracy, or solidity of the contracts. In no event will ETHER AUTHORITY or its partners, employees, agents, or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions or actions with regards to the information provided in this audit report.

The assessment services provided by ETHER AUTHORITY are subject to dependencies and are under continuing development. You agree that your access 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 with high levels of technical risk and uncertainty. The assessment reports could include false positives, negatives, and unpredictable results. The services may access, and depend upon, multiple layers of third parties.