Cyberscope

Audit Report

SquadSwap

November 2023

Repository https://github.com/Bit5Tech

Projects SquadSwap, SquadToken, SquadSwap-v3

Audited by © cyberscope

Table of Contents

Review

Repository	https://github.com/Bit5Tech/SquadSwap-v3
SquadSwap Commit	314650989393387d4b5406de1a2dc718cf907a76
SquadToken Commit	e57260bffc2047718a7d222267e5e5b417ac0a77
SquadSwap-v3 Commit	4b84122cb1116545be78851ae263dc11a63a499c

Audit Updates

Initial Audit	24 Nov 2023

Source Files

Filename	SHA256
SquadToken/contracts/Squad.sol	169cecf842648d2b0ca8fb0548e81022b b7f91ae0f8ee4ea1bca1da70dc91c4a
v2/router/SquadswapRouter02.sol	f86a5b1fc8cbd82ed687b07a56af41681 19d2293635c3bde05cf96367cdabad8
v2/router/libraries/SquadswapLibrary.sol	5691ee36d4c8af7ef0ab33337f1ff39248 d35febef011741735e607fa15f02ec
v2/router/libraries/SafeMath.sol	bee306735bcb43dd72a3fb383ed9fadaa cb26488ccb234c594a300be437e4da7
v2/router/interfaces/IWETH.sol	6526a493132e42947f6dc9c0ffc317f239 b88d9710b72299e0c1ede16cc04690
v2/router/interfaces/ISquadswapRouter02.sol	8fddd314d623aee56861a148092cdb38 a9bc3b8d86a8c95caa349ffe8509a66a
v2/router/interfaces/ISquadswapRouter01.sol	14f7f4481559eb013300a196d180dc231 cea7f441a924c0a5748845152562147
v2/router/interfaces/ISquadswapMigrator.sol	887ed855637f5981cb055b30cebce426 4e8fb31172cc25eae5b20c0197f7a47d
v2/router/interfaces/IERC20.sol	49b7bee5c2f36e6ed17b17c9cb88712b 1298a0641b92c4131ec75a2b5fa513ea
v2/router/interfaces/V1/IUniswapV1Factory.sol	7f4c046f7d6418462afe3bcad6354d30b ca4c54910ee9ccffb54dbc5d52384b9
v2/router/interfaces/V1/IUniswapV1Exchange.sol	abc602a33a18b2cfaed2c0805b8f2b5c7 214064913d71318241ebfd16ec19d44
contracts/factory/SquadswapPair.sol	104c878d3d984ab651ccf4eb54e6ac19 2d35e9db4820e840020a7a23e0c72936
contracts/factory/SquadswapFactory.sol	ea7454d84a8ed0f299f65833295b80924 7e45ac1a780afe38656fbf77ec82d0b

contracts/factory/SquadswapERC20.sol	05c37552868b105aa05265a467b38b75bf2cbf53838c9ac9f1521e857cd36134
contracts/factory/interfaces/ISquadswapPair.sol	a84f0879ff945d65529014b06db1ecbe1e235f84563380be19edd9e076c04979
contracts/factory/interfaces/ISquadswapFactory.sol	54637b75a6def8dc326fae32a813ad60d6908d47af890644917b32427481375c
contracts/factory/interfaces/ISquadswapERC20.sol	d3261a0d4a04943b69ba9786e8320371246d0836f0633858d5f419c8633e1d71
contracts/factory/interfaces/ISquadswapCallee.sol	ddff25e7cd4420f9387e3353e3e5a4df1009e7e340974a7ef8e89ab8f11b37a0
contracts/factory/interfaces/IERC20.sol	2b63f199f838028184efefbcfd6cf2b9192624c3dae5dc1116ecbb15c36a67e8
/v3/v3-periphery/contracts/V3Migrator.sol	b08d9a41f2d9c01e994c2fa292b7cbcdcb33b2d254c10d8ae4cc068eaa2a9fcd
/v3/v3-periphery/contracts/SwapRouter.sol	6ca6c830990aea47fed8b728e87c6161d2bc463e0292b7ea2ea9f080e25c4183
/v3/v3-periphery/contracts/NonfungibleTokenPositionDescriptorOffChainV2.sol	9041d1e442dd614203d15079de17c3a2de449930b65f4a732db0bf1893f382ff
/v3/v3-periphery/contracts/NonfungibleTokenPositionDescriptorOffChain.sol	a8085b77a34122dae1358e8dff09ed0bb91dc84e7ca52a6c1548abe990d9229a
/v3/v3-periphery/contracts/NonfungibleTokenPositionDescriptor.sol	2a8f4a8eb154c9c8422f479b41c760c3d9bb05e99b9c6b10d4497f92e58d4a59
/v3/v3-periphery/contracts/NonfungiblePositionManager.sol	ca9f4b4c105e10c86a15ed14e0598d057580282a82840a191937d3fd9ba47622
/v3/v3-periphery/contracts/NFTDescriptorEx.sol	40ff1c73bf6ee2160ef6a9801711daffc1cc33e0315ae9f8afd0b8ec6957efe8
/v3/v3-periphery/contracts/lens/TickLens.sol	4bae63e0531dcfa4afad4d201866c1343dd588d92aa3b8794fe7dcb4dff5c4a9

/v3/v3-periphery/contracts/lens/SquadInterfaceMulticall.sol	672ac1a6f05c3afe8a379010312305d805cc605c017cf7e3054222cfbd1cef52
/v3/v3-periphery/contracts/lens/QuoterV2.sol	d7284911ce6d2cd64d2244d293f779e308f458a19091abbb034d6f1c7f280542
/v3/v3-periphery/contracts/lens/Quoter.sol	276e9393b48312e0fecf78390e66aeedf7d8b4ee1167241f8348ffecd1be15c4
/v3/v3-periphery/contracts/base/SelfPermit.sol	48bb499a5e2bb8063788faf42ba0abd71cbd63392aa4d4c12531b530419d6afa
/v3/v3-periphery/contracts/base/PoolInitializer.sol	206b365e1e857f08cd6ddfdb92eb40005ee7f66188f3e12a7910c9455fa173f7
/v3/v3-periphery/contracts/base/PeripheryValidation.sol	40877c212ebd04f41a3c582bbbf8ad925f31b2a4f7f129352f55777c8fd584a0
/v3/v3-periphery/contracts/base/PeripheryPaymentsWithFee.sol	6ef994f772d796f196aa7491c684b443c72a5dee223a3f04b0afbbbe82319c65
/v3/v3-periphery/contracts/base/PeripheryPayments.sol	68cef83e01906a13f4a2bb1c12a9e99fad3e957eea6ddbb54bac30ba3b06a436
/v3/v3-periphery/contracts/base/PeripheryImmutableState.sol	f4611f54f13d0599648bf88fc5bba7fe8eb3bfc27f898c5cc0e2f27272ebca99
/v3/v3-periphery/contracts/base/Multicall.sol	029ad0bcade48ff32da51094a3fb245fd7d8324c4fb4dd20fb4b2614efc9618c
/v3/v3-periphery/contracts/base/LiquidityManagement.sol	98922b617bd5221dd505f305fb1484bde4a72d1b7767d18874b6195caf1f67df
/v3/v3-periphery/contracts/base/ERC721Permit.sol	d917dd488471948d666b4c929f9df7a3b4133db6874de2c8c2a1a2e713c0e984
/v3/v3-periphery/contracts/base/BlockTimestamp.sol	e5ca9a8b6b9e0cafcd9a9966b05228a1572f82fccee396d2e0eff5f8aa9bb1f4

/v3/v3-lm-pool/contracts/SquadV3LmPoolDeployer.sol	9cb1703b0d04d9b6429e8bc5c535103600e38ccedb373b711147d0bab589ac79
/v3/v3-lm-pool/contracts/SquadV3LmPool.sol	d91c7edaf01c65c5e40d0ffe5e96d22a521df51413792a10b2e51755e3d219a6
/v3/v3-lm-pool/contracts/libraries/LmTick.sol	99e1fa722381a8e7d9fbc3502804b37bdd1d64fa40965cdcb0681bd50fbcbc42
/v3/v3-lm-pool/contracts/interfaces/ISquadV3LmPool.sol	6c805e3b27a557e157df823712c7a0de13961fa0a565c5f7dceb89c878a6a8bc
/v3/v3-lm-pool/contracts/interfaces/IMasterChefV3.sol	9e91da111914623d4773f23bc08e8b6b552b5d215e415822721d77dd8773027f
/v3/v3-core/contracts/SquadV3PoolDeployer.sol	ecd80fb2a07fcac1e2a79f5ed3c4349ddfcf341948f0dd09cd15e3c70e2e3740
/v3/v3-core/contracts/SquadV3Pool.sol	154e13fc77c65722ea732896df044c79d04c40f2873e7de2bc1605b74e8a7d38
/v3/v3-core/contracts/SquadV3Factory.sol	6bfe0bc1962aa5eb465f9f882552d9cfaa85cf438aa9efcb529f14737fbaa192
/v3/v3-core/contracts/FeeManager.sol	f08689c2474cc5e7629eeae0ce1d33b4e1e37d8233be0a4dd0b280aac6d9bce7
/v3/router/contracts/V3SwapRouter.sol	ec0f4046c3d9a8c142382caa3ed103d46872cfef5fffc7eba449f2495e03aff9
/v3/router/contracts/V2SwapRouter.sol	4e226c6e2cc92f5eeb8cf4947b7d40ee68156576e90477452874469b3bdfb1b0
/v3/router/contracts/StableSwapRouter.sol	920b5b519421033ecbc8a491821e5cda35be18b04e31521b266d192c963aa940
/v3/router/contracts/SmartRouter.sol	ddf30d7215771dcd331e92c0fe23b1a15e493c58cd5af5e6e9f61d6ca2ae87b2
/v3/router/contracts/lens/TokenValidator.sol	5dcd6076e953e861670f81247e1c792f9a16825b19237c80ffbf6facc3e8ffb9

/v3/router/contracts/lens/QuoterV2.sol	91daac85d8bd75570aeacbc659821822d900c1934e4d2e1a567cd5a373ff717e
/v3/router/contracts/lens/Quoter.sol	3fbcdf6cdb21415c45934d6ac65de153f9c960cf4828c46eca95b99d8ec564ff
/v3/router/contracts/lens/MixedRouteQuoterV1.sol	c6dd970defc9daedcbd11e45a9a45cc34cf5cb1947564ad6b025e01f03e656f6
/v3/router/contracts/base/PeripheryValidationExtended.sol	b37fa9da2dfcb07cc3f770bd82d6ab98d77fb5b0fa1ee964c9cbebf3f346355f
/v3/router/contracts/base/PeripheryPaymentsWithFeeExtended.sol	0641943002b9a7cbc5966b3451092af5f5eacdf2847c50d3bcfc2ba5aba49e35
/v3/router/contracts/base/PeripheryPaymentsExtended.sol	52e6bfc69f61b3de72a06a3e8c777a30c28d5cf0a5ae3e88ef00c0fd90d1c5c8
/v3/router/contracts/base/OracleSlippage.sol	dbec16b0de188bc15ce8c8a2a8038bf58d615581c9d355afe7a2191641475968
/v3/router/contracts/base/MulticallExtended.sol	27f61135c65d78fe5dbbde571335947606823d573519b5f412e9414d34a79f5b
/v3/router/contracts/base/ImmutableState.sol	c6bfa945ff513fa64ac9aeb6340251a14b6268ae0df0485de685a6195cb31a55
/v3/router/contracts/base/ApproveAndCall.sol	638e0a5767187270b08c6873c24896f804922caccfadc6ae4c64f24165d22231
/v3/masterchef-v3/test/TestLiquidityAmounts.sol	b6bc4d984ae29d96302c1ed9799d65360d987da2c28f1c4eb287073e9a164ac4
/v3/masterchef-v3/contracts/MasterChefV3.sol	120678ea48e82cf26a15b78885607a9e683d59b110f54c996c9bb606c0a7aecf
/v3/masterchef-v3/contracts/Enumerable.sol	b07a199e4befd5186d6e5d6307ffba3b09b1ae8e6b78549ae41dc37a8c714aca
/v3/masterchef-v3/contracts/utils/Multicall.sol	24945f705c61471f6338630710b81589d5a106dc74e42c0db71fc32a7358b585

/v3/masterchef-v3/contracts/receiver/MasterChefV3ReceiverV2.sol	02eebcb13f077c306374bfeb216557a5bf50e4fd852917dd0eafe9fdfa43069b
/v3/masterchef-v3/contracts/receiver/MasterChefV3Receiver.sol	7e30151ac495e65c3a363ee62b4e68dd36d5284d31e4c863126b4a5485bc5718
/v3/masterchef-v3/contracts/libraries/SafeCast.sol	308851a754c1b946d5664f11db11d622e30bfd3d35015186294bfc3524535644
/v3/masterchef-v3/contracts/keeper/MasterChefV3KeeperV2.sol	ae4bbd773d8f3fa454e2e5da1a7776375402128d62c489a06d0633ba5c2d6aab
/v3/masterchef-v3/contracts/keeper/MasterChefV3KeeperV1.sol	d01d596bace8c03aea1bf790861a3c8a8dc3bbc8cc12c25505f26dc0255b911c
/v3/masterchef-v3/contracts/interfaces/IWETH.sol	567fe8f049b32852d5965ccff17469c641bf9ca7e3393e3dadd27460870cd5c1
/v3/masterchef-v3/contracts/interfaces/ISquadV3Pool.sol	bbb6f6cfda3c09365424d56349dc7ecbefdf6a8237998094d41d09a39d6d960d
/v3/masterchef-v3/contracts/interfaces/IReceiver.sol	5924dcee13f4f189f20a0a31148fe6bff3c051998933ff9127a9e9907c36256f
/v3/masterchef-v3/contracts/interfaces/INonfungiblePositionManagerStruct.sol	b5e107cc467ad2b47aa417ffbaef8659c613cb413a428d6c8ed442e4ac14018f
/v3/masterchef-v3/contracts/interfaces/INonfungiblePositionManager.sol	c6d53cf8abbf3f7a7f246ea98e3332ecd854b88ff31a2962a5e7b13b40b8efa8
/v3/masterchef-v3/contracts/interfaces/IMasterChefV3.sol	640fa673574306a208772ca22010e38ce2079b3cee7304a75173c28b0bccd8c7
/v3/masterchef-v3/contracts/interfaces/IMasterChefV2.sol	94ec05df38a733410f7b2291048604b40271f5276442a1ee39a07921247ae96a
/v3/masterchef-v3/contracts/interfaces/ILMPoolDeployer.sol	fbbca9e76f52c39c024481bbe1a627d4408fd8f3eae21986a70266d590ba6bcf

/v3/masterchef-v3/contracts/interfaces/ILMPool.sol	d6d2deb9b01ca89c110b9f802d42b9fe377284a27018e626a7153837c0da7fc7
/v3/masterchef-v3/contracts/interfaces/IFarmBooster.sol	fb99be458b9adfe9c3b51ff5e0bf77795837b2acc8e327244c6d3ff62ceedb52

Overview

SquadToken

The `SquadToken` contract, is an implementation of a BEP20 token on the Binance Smart Chain (BSC). This contract adheres to the standards set by the BEP20 protocol, ensuring compatibility and functionality within the BSC ecosystem. The contract is structured to provide a secure and efficient means of creating and managing digital assets, leveraging the robustness of blockchain technology. It includes essential features such as token transfer, balance queries, and allowance management, which are fundamental to the operation of any digital token on a blockchain network.

At the core of the SquadToken contract is the implementation of key functionalities that define its behavior and utility. The contract includes mechanisms for ownership management, allowing the initial deployer of the contract to be designated as the owner. This ownership can be transferred or renounced, providing flexibility and control over the contract's administration. Additionally, the contract incorporates the SafeMath library, a critical component for ensuring safe arithmetic operations, thereby mitigating risks such as overflow and underflow errors. This inclusion is particularly important in the context of financial transactions and token management, where accuracy and security are paramount.

Furthermore, the SquadToken contract is designed with user-centric features that enhance its usability within the BSC network. It supports standard BEP20 functions like transferring tokens between accounts, approving third parties to spend tokens on behalf of the token holder, and querying token balances. These functions are integral to the token's interaction with other contracts and users on the network. The contract's adherence to the BEP20 standard ensures that it can seamlessly integrate with a wide range of decentralized applications (dApps) and services within the Binance Smart Chain ecosystem, making it a versatile and valuable asset for various blockchain-based applications.

V2 Contracts

The SquadSwap contracts, stand as a decentralized finance (DeFi) application on the blockchain. They are designed for automated token exchange, leveraging a suite of smart contracts, each renamed to align with the SquadSwap theme. The core contracts include `SquadswapERC20.sol`, which serves as a template for ERC20 tokens within the ecosystem, and `SquadswapFactory.sol`, a central component for creating new liquidity pairs for any two tokens. Additionally, the `SquadswapPair.sol` contract represents individual liquidity pools for token pairs, handling key functionalities like liquidity provision and token swaps.

The platform's architecture is further supported by a range of interfaces and utility contracts. These include `IERC20.sol` and `IBEP20.sol`, ensuring compatibility with a wide range of tokens. The `ISquadswapCallee.sol` interface allows for the implementation of custom logic in response to token transfers. The platform also includes contracts like `SquadswapRouter02.sol` for facilitating multi-step transactions.

Moreover, SquadSwap incorporates various utility and safety contracts such as `Math.sol`, `SafeMath.sol`, and `UQ112x112.sol`, crucial for precise financial calculations and preventing common vulnerabilities. The ecosystem is rounded off with testing and validation contracts like `DeflatingERC20.sol` and `SquadswapERC20Test.sol`, ensuring the platform's functionality and security.

Overall, SquadSwap offers a comprehensive DeFi solution, enabling seamless token swaps, liquidity provision, and a range of other financial activities in a decentralized and secure environment.

V3 Contracts

SquadSwap's adaptation of the V3 protocol introduces a sophisticated and flexible DeFi platform on the blockchain, tailored to offer enhanced liquidity and trading features.

At the base of this ecosystem is the `SquadV3Factory.sol`, a pivotal contract responsible for creating and managing liquidity pools, known as SquadV3Pools. These pools, defined in `SquadV3Pool.sol` and `SquadV3PoolDeployer.sol`, are where liquidity providers can add their assets and traders can swap tokens. The unique feature of these pools is their concentrated liquidity, allowing liquidity providers to allocate their capital within specific price ranges, optimizing capital efficiency.

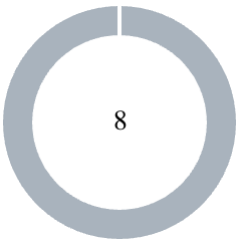
The `NonfungiblePositionManager.sol` and `NonfungibleTokenPositionDescriptor.sol` contracts play a crucial role in managing liquidity positions, which are represented as NFTs. This approach allows for more granular control and flexibility over individual liquidity positions. The `SwapRouter.sol` and `ISwapRouter.sol` contracts facilitate the actual token swaps, offering an interface for users to easily trade tokens within the SquadSwap ecosystem.

Safety and utility are also paramount in this ecosystem. Contracts like `SafeCast.sol`, `TickMath.sol`, and `LowGasSafeMath.sol` ensure accurate and efficient mathematical operations, crucial for financial transactions. The `FeeManager.sol` and `PeripheryPaymentsWithFee.sol` contracts handle the platform's fee structure and distribution, ensuring a fair and sustainable system.

Furthermore, the platform includes advanced features like the `V3Migrator.sol` for migrating liquidity from previous versions, and the `MasterChefV3.sol` for incentivizing liquidity provision. The integration of `IWETH.sol` and `IWETH9.sol` ensures seamless interaction with the wrapped token, a key aspect of operating on the blockchain.

In summary, SquadSwap's implementation of the UniswapV3 protocol on the blockchain offers a robust and feature-rich DeFi platform. It provides users with advanced trading and liquidity provision options, enhanced by the concentrated liquidity feature, while ensuring security and efficiency through a suite of carefully designed smart contracts.

Findings Breakdown



Critical	0
Medium	0
Minor / Informative	8

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	8	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	RSML	Redundant SafeMath Library	Unresolved
	L09	Dead Code Elimination	Unresolved
	L19	Stable Compiler Version	Unresolved
	L04	Conformance to Solidity Naming Conventions	Unresolved
	L13	Divide before Multiply Operation	Unresolved
	L16	Validate Variable Setters	Unresolved
	L17	Usage of Solidity Assembly	Unresolved
	L18	Multiple Pragma Directives	Unresolved

RSML - Redundant SafeMath Library

Criticality	Minor / Informative
Location	Squad.sol
Status	Unresolved

Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

L09 - Dead Code Elimination

Criticality	Minor / Informative
Location	Squad.sol#L723
Status	Unresolved

Description

In Solidity, dead code is code that is written in the contract, but is never executed or reached during normal contract execution. Dead code can occur for a variety of reasons, such as:

- Conditional statements that are always false.
- Functions that are never called.
- Unreachable code (e.g., code that follows a return statement).

Dead code can make a contract more difficult to understand and maintain, and can also increase the size of the contract and the cost of deploying and interacting with it.

```
function getChainId() internal view returns (uint) {
        uint256 chainId;
        assembly { chainId := chainid() }
        return chainId;
    }
```

Recommendation

To avoid creating dead code, it's important to carefully consider the logic and flow of the contract and to remove any code that is not needed or that is never executed. This can help improve the clarity and efficiency of the contract.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	Squad.sol#L5
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.0;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	contracts/factory/SquadswapPair.sol#L66contracts/factory/SquadswapFactory.sol#L42,47contracts/factory/SquadswapERC20.sol#L16contracts/factory/interfaces/ISquadswapPair.sol#L18,19,36contracts/factory/interfaces/ISquadswapERC20.sol#L18,19
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address _token0
address _token1
address _feeTo
address _feeToSetter
bytes32 public DOMAIN_SEPARATOR
function DOMAIN_SEPARATOR() external view returns (bytes32);
function PERMIT_TYPEHASH() external pure returns (bytes32);
function MINIMUM_LIQUIDITY() external pure returns (uint);
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation
https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L13 - Divide before Multiply Operation

Criticality	Minor / Informative
Location	contracts/factory/SquadswapPair.sol#L100
Status	Unresolved

Description

It is important to be aware of the order of operations when performing arithmetic calculations. This is especially important when working with large numbers, as the order of operations can affect the final result of the calculation. Performing divisions before multiplications may cause loss of prediction.

```
uint liquidity = numerator / denominator * 6 * 4 / 5
```

Recommendation

To avoid this issue, it is recommended to carefully consider the order of operations when performing arithmetic calculations in Solidity. It's generally a good idea to use parentheses to specify the order of operations. The basic rule is that the multiplications should be prior to the divisions.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	contracts/factory/SquadswapPair.sol#L68,69contracts/factory/SquadswapFactory.sol#L18,44,49
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
token0 = _token0
token1 = _token1
feeToSetter = _feeToSetter
feeTo = _feeTo
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L17 - Usage of Solidity Assembly

Criticality	Minor / Informative
Location	contracts/factory/SquadswapFactory.sol#L32contracts/factory/SquadswapERC20.sol#L26
Status	Unresolved

Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
            pair := create2(0, add(bytecode, 32), mload(bytecode), salt)
        }

assembly {
            chainId := chainid
        }
```

Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

L18 - Multiple Pragma Directives

Criticality	Minor / Informative
Location	contracts/factory/SquadswapPair.sol#L1contracts/factory/SquadswapFactory.sol#L1contracts/factory/SquadswapERC20.sol#L1contracts/factory/libraries/UQ112x112.sol#L1contracts/factory/libraries/SafeMath.sol#L1contracts/factory/libraries/Math.sol#L1contracts/factory/interfaces/ISquadswapPair.sol#L1contracts/factory/interfaces/ISquadswapFactory.sol#L1contracts/factory/interfaces/ISquadswapERC20.sol#L1contracts/factory/interfaces/ISquadswapCallee.sol#L1contracts/factory/interfaces/IERC20.sol#L1
Status	Unresolved

Description

If the contract includes multiple conflicting pragma directives, it may produce unexpected errors. To avoid this, it's important to include the correct pragma directive at the top of the contract and to ensure that it is the only pragma directive included in the contract.

```
pragma solidity =0.5.16;
pragma solidity >=0.5.0;
```

Recommendation

It is important to include only one pragma directive at the top of the contract and to ensure that it accurately reflects the version of Solidity that the contract is written in.

By including all required compiler options and flags in a single pragma directive, the potential conflicts could be avoided and ensure that the contract can be compiled correctly.

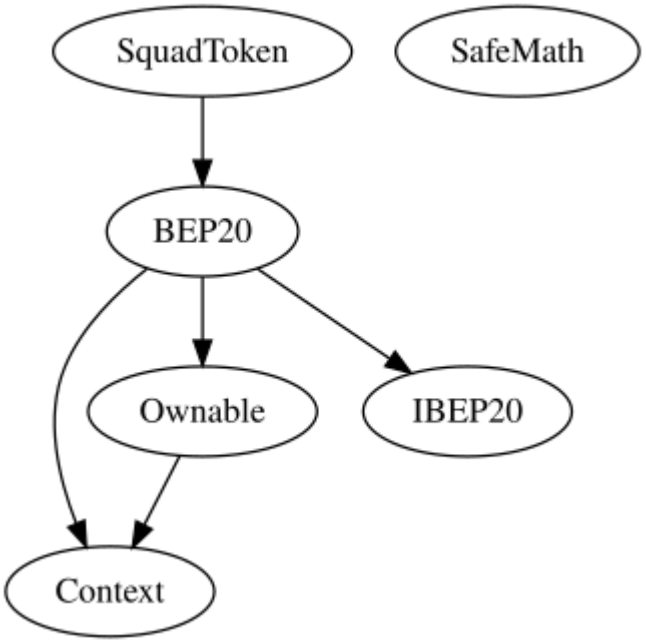
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
	_msgSender	Internal		
Ownable	Implementation	Context		
		Public	✓	-
	owner	Public		-
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
IBEP20	Interface			
	totalSupply	External		-
	decimals	External		-
	symbol	External		-
	name	External		-
	getOwner	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-

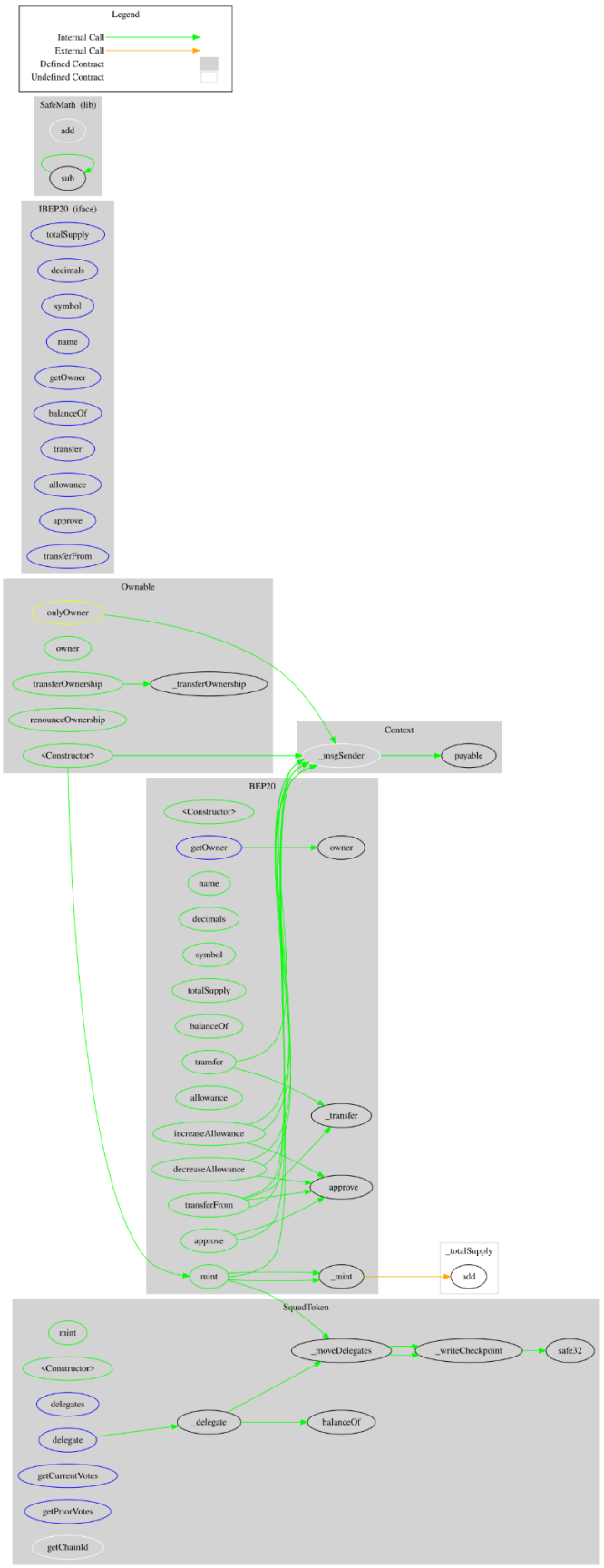
	approve	External	✓	-
	transferFrom	External	✓	-
SafeMath	Library			
	add	Internal		
	sub	Internal		
	sub	Internal		
BEP20	Implementation	Context, IBEP20, Ownable		
		Public	✓	-
	getOwner	External		-
	name	Public		-
	decimals	Public		-
	symbol	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	transfer	Public	✓	-
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	mint	Public	✓	onlyOwner

	_transfer	Internal	✓	
	_mint	Internal	✓	
	_approve	Internal	✓	
SquadToken	Implementation	BEP20		
	mint	Public	✓	onlyOwner
		Public	✓	-
	delegates	External		-
	delegate	External	✓	-
	getCurrentVotes	External		-
	getPriorVotes	External		-
	_delegate	Internal	✓	
	_moveDelegates	Internal	✓	
	_writeCheckpoint	Internal	✓	
	safe32	Internal		
	getChainId	Internal		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
SafeMath (lib)
add
sub
IBEP20 (iface)
totalSupply
decimals
symbol
name
getOwner
balanceOf
transfer
allowance
approve
transferFrom
Ownable
onlyOwner
owner
transferOwnership
_transferOwnership
renounceOwnership
<Constructor>
Context
_msgSender
payable
BEP20
<Constructor>
getOwner
owner
name
decimals
symbol
totalSupply
balanceOf
transfer
allowance
increaseAllowance
decreaseAllowance
transferFrom
approve
mint
_transfer
_approve
_mint
_totalSupply
add
SquadToken
mint
<Constructor>
delegates
delegate
getCurrentVotes
getPriorVotes
getChainId
_delegate
_moveDelegates
_writeCheckpoint
safe32
balanceOf

Summary

SquadSwap contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io