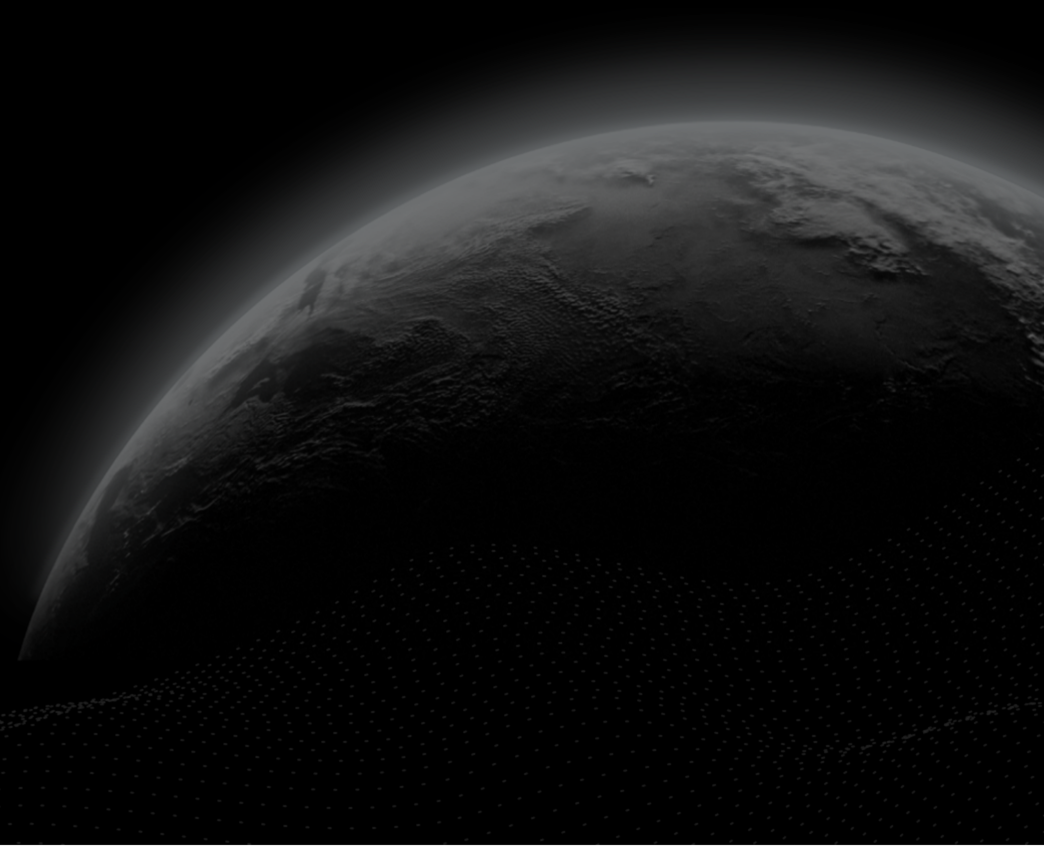
CERTIK

Security Assessment

Playbunny [EX] - Audit

CertiK Assessed on Oct 18th, 2023



CertiK Assessed on Oct 18th, 2023

Playbunny [EX] - Audit

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
DeFi	EVM Compatible	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 10/18/2023	N/A

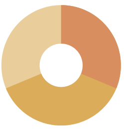
CODEBASE
karma-tokens
View All in Codebase Page

COMMITS
dd141161cdb1a2bc1cad51fae74cc30be06d804f
View All in Codebase Page

Highlighted Centralization Risks

- Contract upgradeability
- Privileged role can mint tokens
- Transfers can be paused
- Fees are bounded by 15%

Vulnerability Summary



16	11	0	0	5	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

Count	Severity	Status	Description
0	Critical		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.
5	Major	2 Resolved, 3 Acknowledged	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.
6	Medium	6 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
5	Minor	3 Resolved, 2 Acknowledged	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.
0	Informational		Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | Playbunny [EX] - AUDIT

CODEBASE | Playbunny [EX] - AUDIT

Repository

karma-tokens

Commit

dd141161cdb1a2bc1cad51fae74cc30be06d804f

AUDIT SCOPE | Playbunny [EX] - AUDIT

16 files audited ● 2 files with Acknowledged findings ● 6 files with Resolved findings ● 8 files without findings

ID	File	SHA256 Checksum
DDT	buyback/DividendDistributor.sol	adab3e9de36f328193f97be6e9a63f88904f9118f6db1d58a22cb522aaa035a3
BTT	BaseToken.sol	3cff9526258b9328aabfcdd3e335481238ccf30040849c4abeef9bb64948a848
RTT	ReflectionToken.sol	df9ac1a7ec8d9f6bc08025a90b6865a448c6b0cd739337b93c005d1225192312
STT	StandardToken.sol	3f49c129f3d1ba60505e4dbd3aed44afc8a564b69d88cc2d5f45f6557a451247
ERE	ERC1363/ERC1363.sol	0cc80eb6d8adfe7cbfec6bd220953b2288ca626dd768e0b3d88cb83736910013
ERR	ERC2612/ERC2612.sol	f94b07fddf1f232757404e14d9fe314e81de1cdcc1d3c457004b8ff094f7255a
ERM	extensions/ERC20Mintable.sol	b20a7dae7b9a044a10129e573d4a5b5d1b71890fc2a676691266e53faf377d5c
ECT	extensions/ERC20TokenRecover.sol	00d7b82e28c33056aa62b7d8dec3228cf8a4078412b3eaf2f2393b330b8fa813
IER	ERC1363/IERC1363.sol	7e7c564ce0d07dacb73d49909ed0bf691fa27c3997fce12cee8aa5ebc759075b
IEC	ERC1363/IERC1363Receiver.sol	266109c0ca8e1bbe84fe0fee7779a580c86198949d62cd58c37cf53b055c9136
IES	ERC1363/IERC1363Spender.sol	935d11f1eb34dc0c5fcbf84162cdc8df55c30a4aa9e6aa81d608656410320ee7
IEE	ERC2612/IERC2612.sol	12c8b1de56868d946920997c387481bbc5c37fffed17dced7d2c0da3330fcc75
IDD	buyback/IDividendDistributor.sol	089623acf967bdf6e53cb4846c45295bd197f843307cd80a2133b080a8bc5602
ERB	extensions/ERC20Burnable.sol	fdba67cd1835948de1b519a161ac29d3c4ae1af1af0f05ebfe9470665950a6aa

ID	File	SHA256 Checksum
ERD	extensions/ERC20Capped.sol	925cb59ccd826ae0a84083319f9979bc8d7d4ff07dac49a03b9ab93d71410e1c
IEM	extensions/IERC20Mintable.sol	821f26c712cf05d13b207a2de506b21c91ac3a3c18f97b508be90a3ec8e5989e

APPROACH & METHODS | Playbunny [EX] - AUDIT

This report has been prepared for Playbunny to discover issues and vulnerabilities in the source code of the Playbunny [EX] - Audit project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

REVIEW NOTES | Playbunny [EX] - AUDIT

File https://github.com/TradersClubDev/karma-tokens/blob/main/contracts/Token.sol is not in original scope of the audit.

Token.sol derived from StandardToken.sol but excluding the transaction fee.

FINDINGS | Playbunny [EX] - AUDIT

16	0	5	6	5	0
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Playbunny [EX] - Audit. Through this audit, we have uncovered 16 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
BTT-01	**Initial Token Distribution**	**Centralization**	**Major**	**Acknowledged**
DDT-04	Potential DOS Attack	Logical Issue	Major	Resolved
GLOBAL-01	**Centralization Related Risks**	**Centralization**	**Major**	**Acknowledged**
GLOBAL-02	**Centralized Control Of Contract Upgrade**	**Centralization**	**Major**	**Acknowledged**
TCD-01	**Centralization Risk Related To `addLiquidityETH`**	**Centralization**	**Major**	**Resolved**
DDT-02	Potential Out-Of-Gas Exception	Gas Optimization, Logical Issue	Medium	Resolved
DDT-03	Unreachable Code	Volatile Code	Medium	Resolved
DDT-08	Potential Reentrancy Attack	Concurrency	Medium	Resolved
RTT-01	Uninitialized Storage Variable	Volatile Code	Medium	Resolved
TCD-02	State Variables In Upgradeable Contracts Are Initialized When Declared	Logical Issue	Medium	Resolved

ID	Title	Category	Severity	Status
TCD-03	Lack Of Storage Gap In Upgradeable Contract	Logical Issue	Medium	Resolved
DDT-05	Potential Sandwich Attack	Financial Manipulation	Minor	Acknowledged
DDT-06	Potential Divide By Zero	Logical Issue	Minor	Resolved
DDT-07	Untrusted Router Address Risk	Logical Issue	Minor	Resolved
ERR-01	Potential Cross-Chain Replay Attack	Logical Issue	Minor	Resolved
GLOBAL-03	Third-Party Dependency Usage	Design Issue	Minor	Acknowledged

BTT-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	Major	BaseToken.sol (v1): 51	Acknowledged

Description

All tokens are sent to the contract deployer when deploying the contract. This is a potential centralization risk as the deployer can distribute tokens without the consensus of the community.

Recommendation

We recommend transparency through providing a breakdown of the intended initial token distribution in a public location. We also recommend the team make an effort to restrict the access of the corresponding private key.

DDT-04 | POTENTIAL DOS ATTACK

Category	Severity	Location	Status
Logical Issue	Major	buyback/DividendDistributor.sol (v1): 160	Resolved

Description

During the execution of the `distributeDividend` function, users' rewards are distributed by invoking the `transfer` function. However, if the reward token is ERC1363 and the user is a malicious smart contract that reverts any payment, the function will be reverted.

Recommendation

External calls can fail accidentally or deliberately, which can cause a DoS condition in the contract. To minimize the damage caused by such failures, it is better to isolate each external call into its own transaction that can be initiated by the recipient of the call.

Alleviation

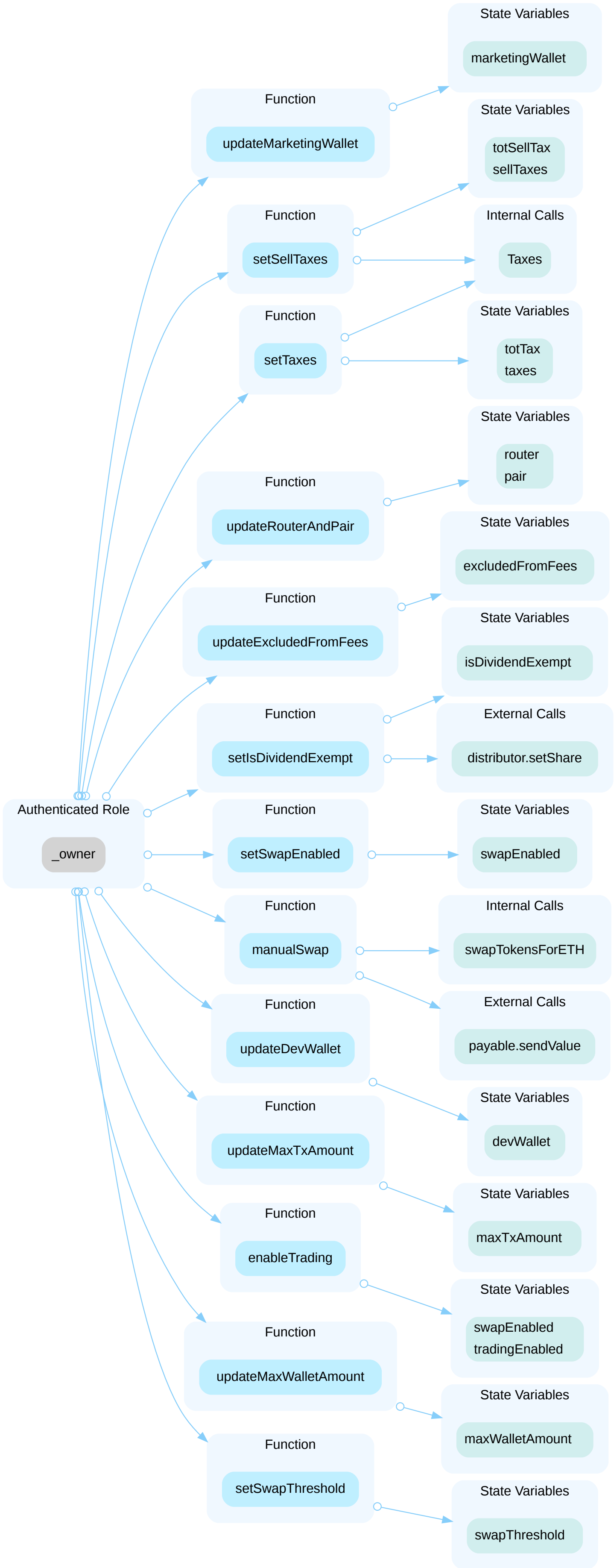
The client revised the code in commit : c0d5478985ba385cae829f59e9b0f56c74666ad9.

GLOBAL-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	Major		Acknowledged

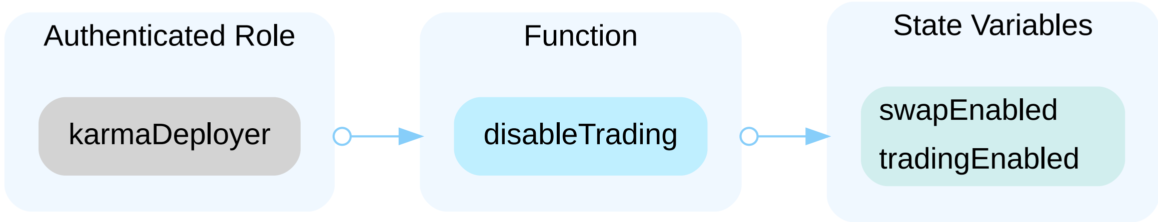
Description

In the contract `ReflectionToken` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.

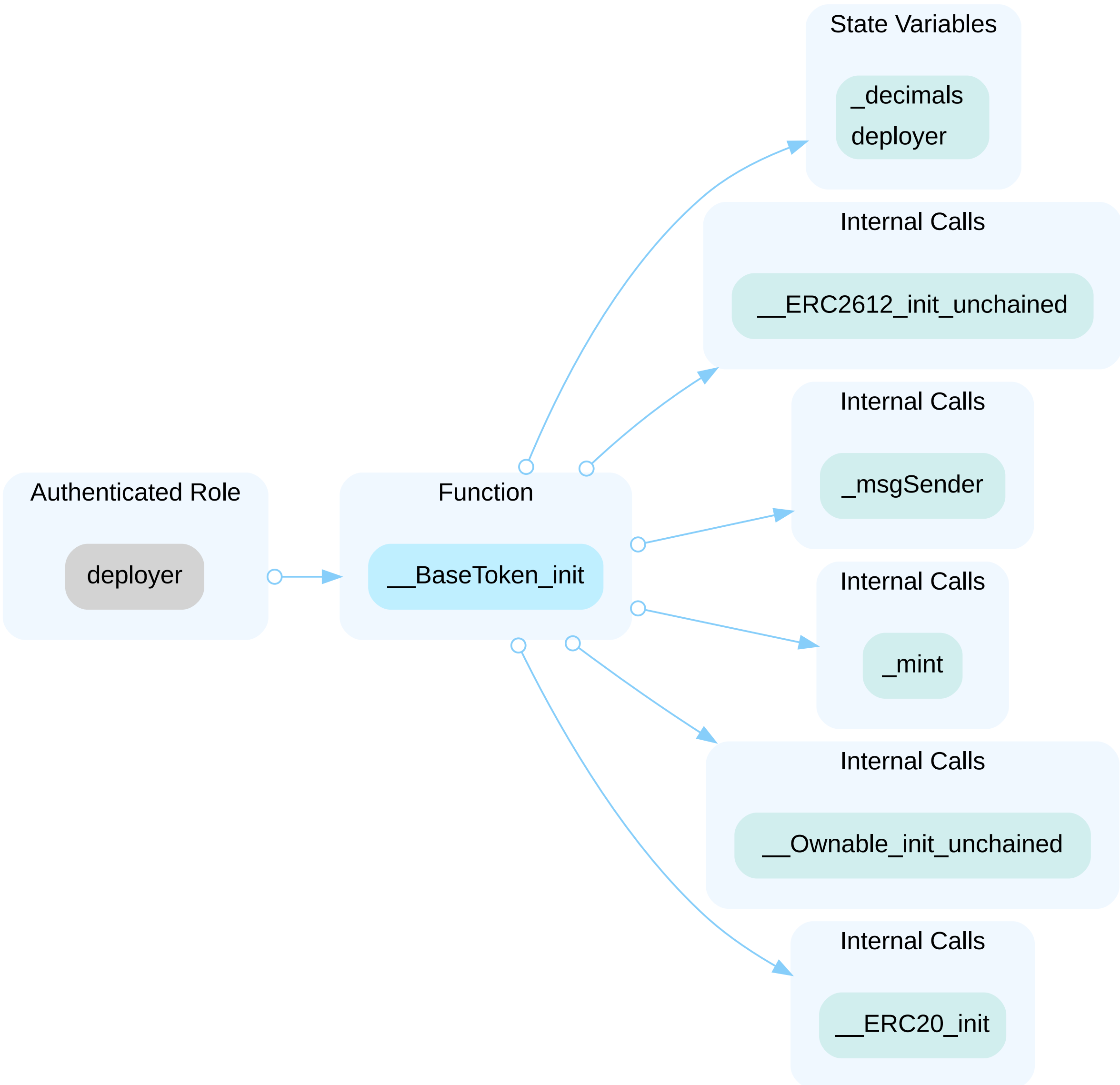


State Variables
marketingWallet
Function
updateMarketingWallet
State Variables
totSellTax
sellTaxes
Function
setSellTaxes
Internal Calls
Taxes
Function
setTaxes
State Variables
totTax
taxes
State Variables
router
pair
Function
updateRouterAndPair
State Variables
excludedFromFees
Function
updateExcludedFromFees
State Variables
isDividendExempt
Function
setIsDividendExempt
External Calls
distributor.setShare
Authenticated Role
_owner
Function
setSwapEnabled
State Variables
swapEnabled
Function
manualSwap
Internal Calls
swapTokensForETH
Function
updateDevWallet
External Calls
payable.sendValue
Function
updateMaxTxAmount
State Variables
devWallet
Function
enableTrading
State Variables
maxTxAmount
State Variables
swapEnabled
tradingEnabled
Function
updateMaxWalletAmount
State Variables
maxWalletAmount
Function
setSwapThreshold
State Variables
swapThreshold

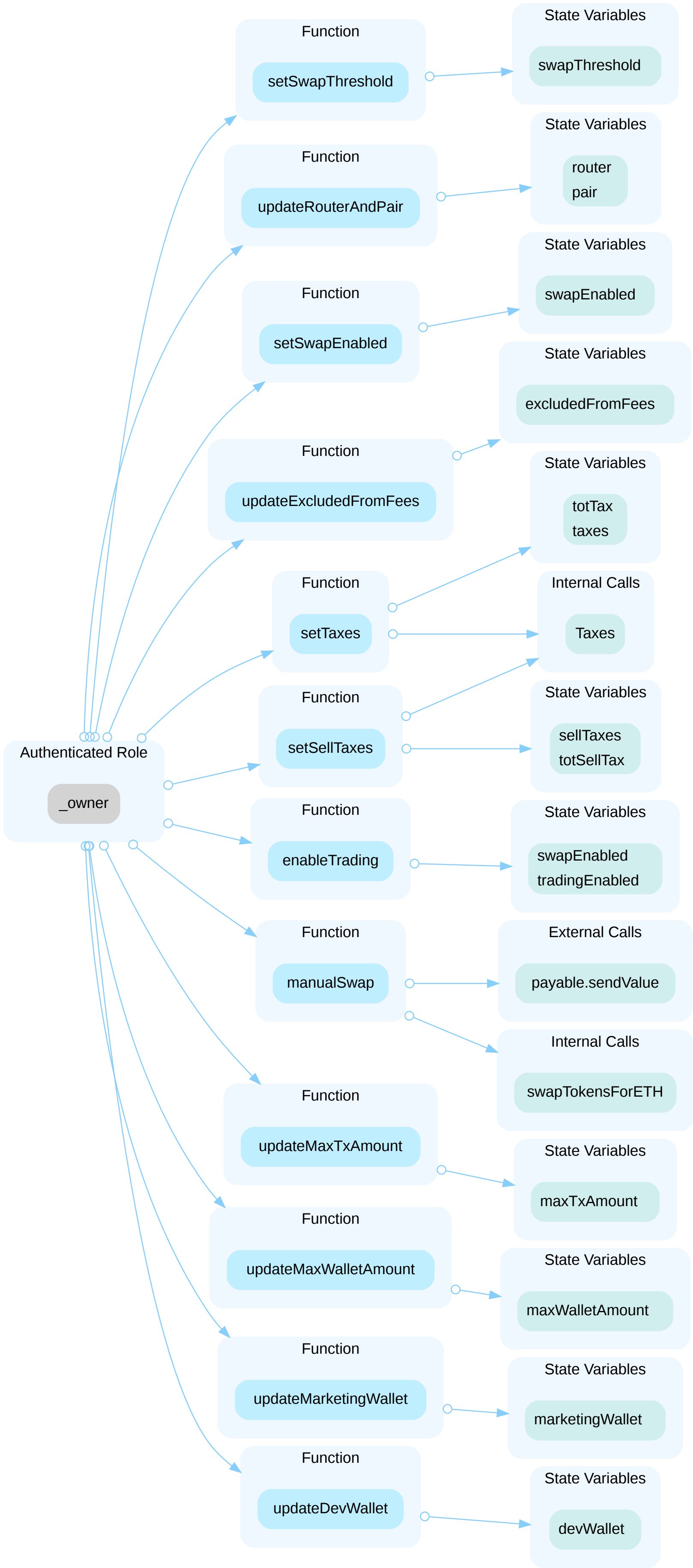
In the contract `ReflectionToken` the role `karmaDeployer` has authority over the functions shown in the diagram below. Any compromise to the `karmaDeployer` account may allow the hacker to take advantage of this authority.



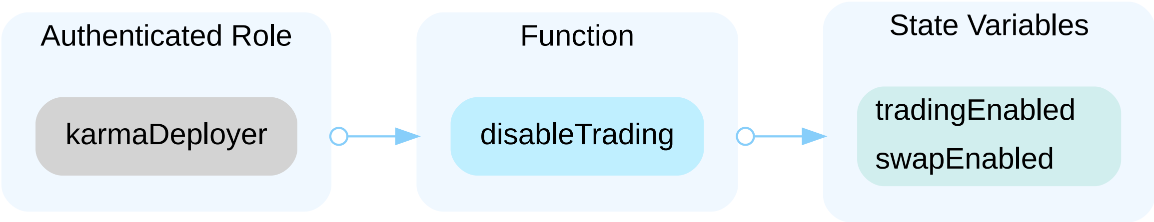
In the contract `BaseToken` the role `deployer` has authority over the functions shown in the diagram below. Any compromise to the `deployer` account may allow the hacker to take advantage of this authority.



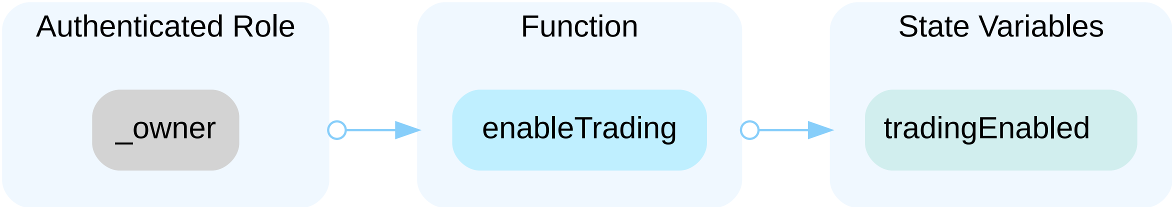
In the contract `StandardToken` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.


Authenticated Role
_owner
Function
setSwapThreshold
State Variables
swapThreshold
Function
updateRouterAndPair
State Variables
router
pair
Function
setSwapEnabled
State Variables
swapEnabled
Function
updateExcludedFromFees
State Variables
excludedFromFees
Function
setTaxes
State Variables
totTax
taxes
Internal Calls
Taxes
Function
setSellTaxes
State Variables
sellTaxes
totSellTax
Function
enableTrading
State Variables
swapEnabled
tradingEnabled
Function
manualSwap
External Calls
payable.sendValue
Internal Calls
swapTokensForETH
Function
updateMaxTxAmount
State Variables
maxTxAmount
Function
updateMaxWalletAmount
State Variables
maxWalletAmount
Function
updateMarketingWallet
State Variables
marketingWallet
Function
updateDevWallet
State Variables
devWallet

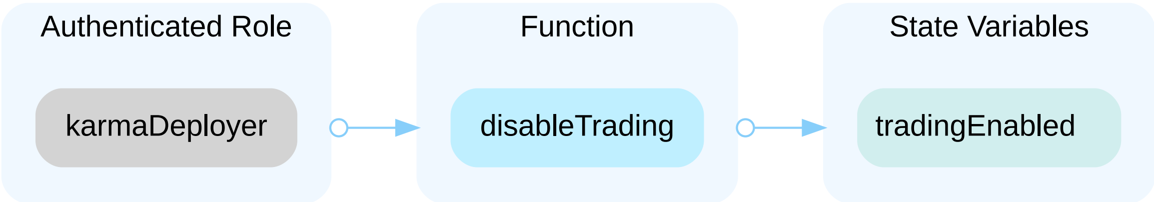
In the contract `StandardToken` the role `karmaDeployer` has authority over the functions shown in the diagram below. Any compromise to the `karmaDeployer` account may allow the hacker to take advantage of this authority.



In the contract `Token` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



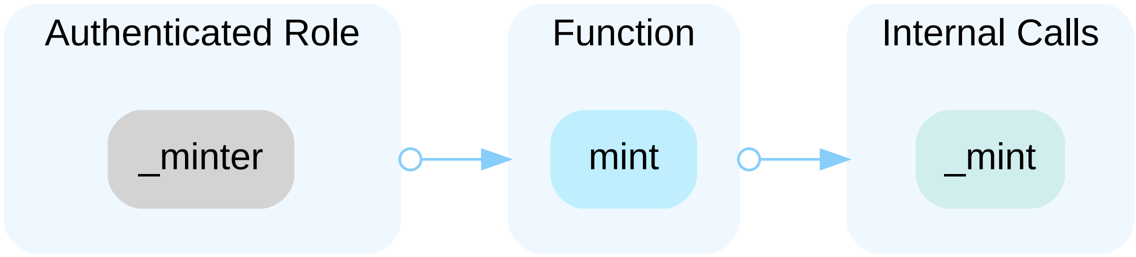
In the contract `Token` the role `karmaDeployer` has authority over the functions shown in the diagram below. Any compromise to the `karmaDeployer` account may allow the hacker to take advantage of this authority.



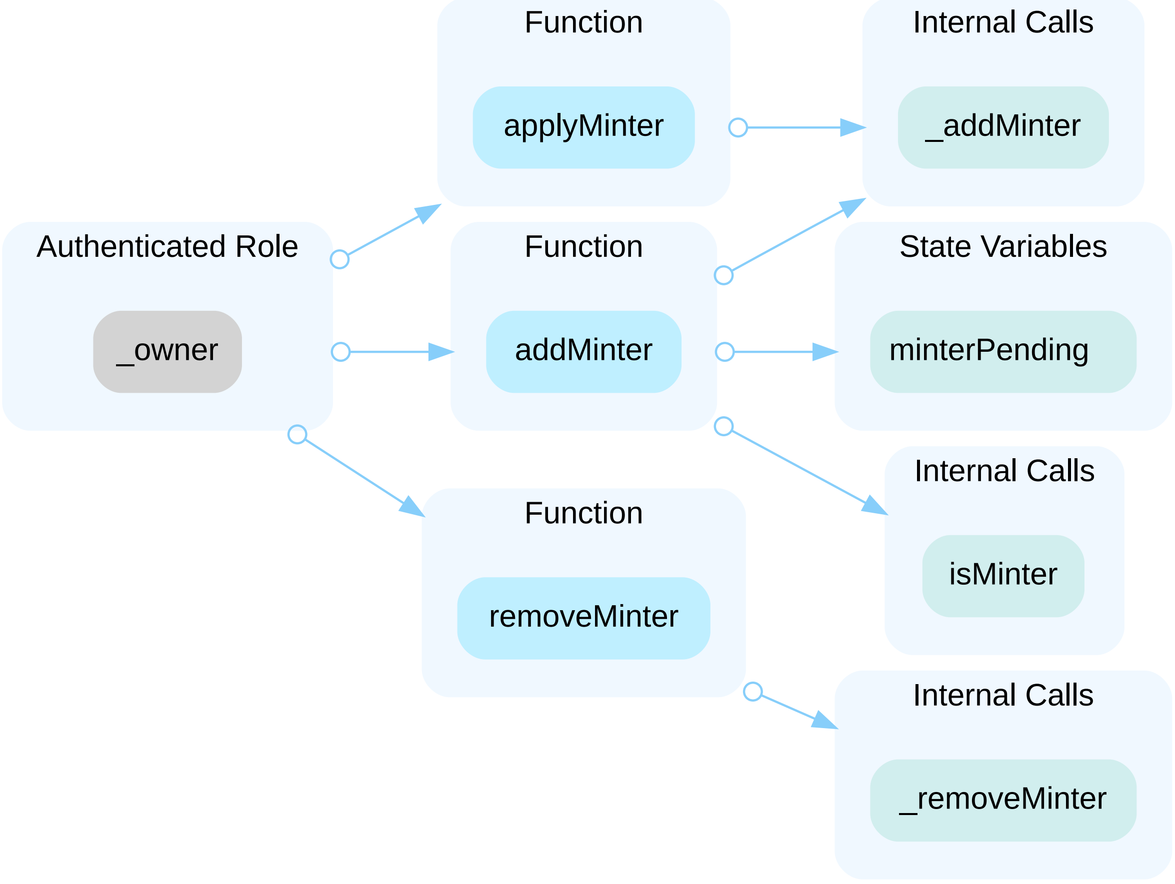
In the contract `DividendDistributor` the role `_token` has authority over the functions shown in the diagram below. Any compromise to the `_token` account may allow the hacker to take advantage of this authority.


External Calls
gasLeft.sub
State Variables
currentIndex
Internal Calls
shouldDistribute
Function
process
External Calls
gasUsed.add
Internal Calls
distributeDividend
State Variables
totalShares
shares
External Calls
totalShares.sub
External Calls
.add
Function
setShare
Internal Calls
addShareholder
Internal Calls
removeShareholder
Internal Calls
getCumulativeDividends
Authenticated Role
_token
External Calls
dividendsPerShareAccuracyFactor.mul
External Calls
router.WETH
External Calls
dividendsPerShare.add
State Variables
totalDividends
dividendsPerShare
Function
deposit
External Calls
router.swapExactETHForTokensSupportingFeeOnTransferTokens
External Calls
rewardToken.balanceOf
External Calls
.div
Function
setDistributionCriteria
External Calls
totalDividends.add
External Calls
.sub
State Variables
minDistribution
minPeriod

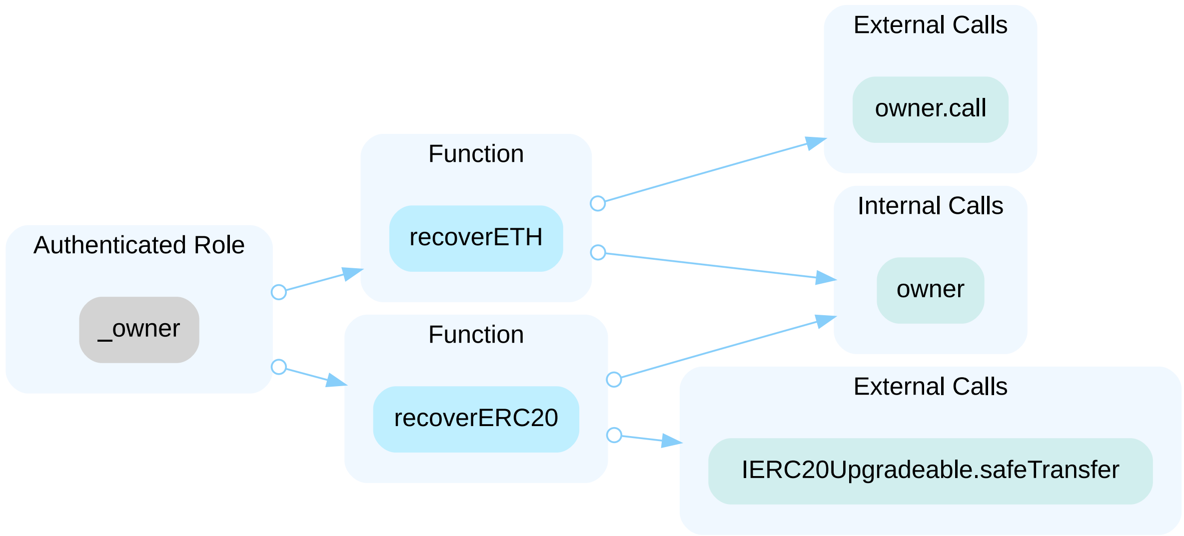
In the contract `ERC20Mintable` the role `_minter` has authority over the functions shown in the diagram below. Any compromise to the `_minter` account may allow the hacker to take advantage of this authority.



In the contract `ERC20Mintable` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



In the contract `ERC20TokenRecover` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority.



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We recommend carefully managing the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term, and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key being compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness of privileged operations;
  AND

- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Alleviation

[CertiK]: The client added a Limited Owner contract in the remediation phase, and making some of the privileged functionalities to be available for owner, limited owner and `karmaCampaignFactory` .

GLOBAL-02 | CENTRALIZED CONTROL OF CONTRACT UPGRADE

Category	Severity	Location	Status
Centralization	● Major		● Acknowledged

Description

In the contract Reflecttoken and Standardtoken, the role `admin` has the authority to update the implementation contract behind the proxy contract.

Any compromise to the `admin` account may allow a hacker to take advantage of this authority and change the implementation contract which is pointed by proxy and therefore execute potential malicious functionality in the implementation contract.

Recommendation

We recommend that the team make efforts to restrict access to the admin of the proxy contract. A strategy of combining a time-lock and a multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise. In addition, the team should be transparent and notify the community in advance whenever they plan to migrate to a new implementation contract.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

Short Term:

A combination of a time-lock and a multi signature (⅔, ⅗) wallet mitigate the risk by delaying the sensitive operation and avoiding a single point of key management failure.

- A time-lock with reasonable latency, such as 48 hours, for awareness of privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromised;
  AND
- A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.

- Provide a link to the **medium/blog** with all of the above information included.

Long Term:

A combination of a time-lock on the contract upgrade operation and a DAO for controlling the upgrade operation mitigate the contract upgrade risk by applying transparency and decentralization.

- A time-lock with reasonable latency, such as 48 hours, for community awareness of privileged operations;
  AND
- Introduction of a DAO, governance, or voting module to increase decentralization, transparency, and user involvement;
  AND
- A medium/blog link for sharing the time-lock contract, multi-signers addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

- Provide the deployed time-lock address.
- Provide the **gnosis** address with **ALL** the multi-signer addresses for the verification process.
- Provide a link to the **medium/blog** with all of the above information included.

Permanent:

Renouncing ownership of the `admin` account or removing the upgrade functionality can *fully* resolve the risk.

- Renounce the ownership and never claim back the privileged role;
  OR
- Remove the risky functionality.

Note: we recommend the project team consider the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

TCD-01 | CENTRALIZATION RISK RELATED TO `addLiquidityETH`

Category	Severity	Location	Status
Centralization	**Major**	**ReflectionToken.sol (v1): 243~248; StandardToken.sol (v1): 206~211**	**Resolved**

Description

`addLiquidity()` uses `devWallet` address as an LP recipient. This allows the `devWallet` to extract all the funds from the AMM pair.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `devWallet` account is compromised.

Alleviation

[**The Karma Pad Team**, 09/28/2023]: The piece of code has been removed.

DDT-02 | POTENTIAL OUT-OF-GAS EXCEPTION

Category	Severity	Location	Status
Gas Optimization, Logical Issue	Medium	buyback/DividendDistributor.sol (v1): 126	Resolved

Description

If the value of `gasleft()` outside the `while` loop is less than the passed parameter `gas` , the `while` loop may throw an `out-of-gas` exception.

```
126         while (gasUsed < gas && iterations < shareholderCount) {
127             if (currentIndex >= shareholderCount) {
128                 currentIndex = 0;
129             }
130
131             if (shouldDistribute(shareholders[currentIndex])) {
132                 distributeDividend(shareholders[currentIndex]);
133             }
134
135             gasUsed = gasUsed.add(gasLeft.sub(gasleft()));
136             gasLeft = gasleft();
137             currentIndex++;
138             iterations++;
139         }
```

Recommendation

We recommend adding a gas verify condition in the `while` loop, so that avoid throw `out-of-gas` exception.

Alleviation

Resolved in commit 0283052f33840c387e82151caf2dc02263dfa3e7.

DDT-03 | UNREACHABLE CODE

Category	Severity	Location	Status
Volatile Code	Medium	buyback/DividendDistributor.sol (v1): 63, 93	Resolved

Description

Functions `deposit()` and `setDistributionCriteria()` in `DividendDistributor` are decorated with the modifier `onlyToken`. Based on our understanding, the token address is contract `ReflectionToken`. However, contract `ReflectionToken` does not call deposit/setDistributionCriteria inside its contract. Therefore, deposit/setDistributionCriteria cannot be called by anyone.

Recommendation

We recommend the team revise the code.

Alleviation

The client revised the code and resolved this issue in commit : dd4525ef44885bde7f8a8c9b7f2c901e9d33b4a3.

DDT-08 | POTENTIAL REENTRANCY ATTACK

Category	Severity	Location	Status
Concurrency	● Medium	buyback/DividendDistributor.sol (v1): 77, 87, 88~90, 128, 132, 137, 160, 162~164, 165~167	● Resolved

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

External call(s)

```
77                distributeDividend(shareholder);
```

- This function call executes the following external call(s).
- In `DividendDistributor.distributeDividend`,
    - `rewardToken.transfer(shareholder,amount)`

State variables written after the call(s)

```
87            shares[shareholder].amount = amount;
```

```
88            shares[shareholder].totalExcluded = getCumulativeDividends(
89                shares[shareholder].amount
90            );
```

External call(s)

```
132                 distributeDividend(shareholders[currentIndex]);
```

- This function call executes the following external call(s).
- In `DividendDistributor.distributeDividend`,
    - `rewardToken.transfer(shareholder,amount)`

State variables written after the call(s)

```
                currentIndex = 0;
```

```
            currentIndex++;
```

External call(s)

```
            rewardToken.transfer(shareholder, amount);
```

State variables written after the call(s)

```
            shares[shareholder].totalRealised = shares[shareholder]
                .totalRealised
                .add(amount);
```

```
            shares[shareholder].totalExcluded = getCumulativeDividends(
                shares[shareholder].amount
            );
```

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

Alleviation

`nonReentrant` is added on function `distributeDividend`.

RTT-01 | UNINITIALIZED STORAGE VARIABLE

Category	Severity	Location	Status
Volatile Code	Medium	ReflectionToken.sol (v1): 182	Resolved

Description

An uninitialized storage variable will contain references to state variables and a critical state variable may be modified unexpectedly.

Recommendation

We recommend initializing all storage variable. For more information on this vulnerability, see SWC-109. of the SWC Registry.

Alleviation

Resolved in commit 1ee2057b63e4d258d63808937e4fa5d4c3f765c1.

TCD-02 | STATE VARIABLES IN UPGRADEABLE CONTRACTS ARE INITIALIZED WHEN DECLARED

Category	Severity	Location	Status
Logical Issue	Medium	BaseToken.sol (v1): 26; extensions/ERC20Mintable.sol (v1): 25	Resolved

Description

State variables initialized when declared are equivalent to initializing them inside the constructor. Therefore, initializing state variables when declared in an upgradeable contract has no actual effect since the constructor of an upgradeable contract is never called.

Recommendation

We recommend initializing state variables in an initializer function if necessary to avoid unexpected behavior and confusion.

Alleviation

The client revised the code and resolved this issue in commit : dd4525ef44885bde7f8a8c9b7f2c901e9d33b4a3.

TCD-03 | LACK OF STORAGE GAP IN UPGRADEABLE CONTRACT

Category	Severity	Location	Status
Logical Issue	Medium	BaseToken.sol (v1): 14; ERC1363/ERC1363.sol (v1): 18; ERC2612/ERC2612.sol (v1): 13; extensions/ERC20TokenRecover.sol (v1): 15	Resolved

Description

There is no storage gap preserved in the logic contract. Any logic contract that acts as a base contract that needs to be inherited by other upgradeable child should have a reasonable size of storage gap preserved for the new state variable introduced by the future upgrades.

Recommendation

We recommend having a storage gap of a reasonable size preserved in the logic contract in case that new state variables are introduced in future upgrades. For more information, please refer to: https://docs.openzeppelin.com/contracts/3.x/upgradeable#storage_gaps.

Alleviation

The client revised the code and resolved this issue in commit : dd4525ef44885bde7f8a8c9b7f2c901e9d33b4a3.

DDT-05 | POTENTIAL SANDWICH ATTACK

Category	Severity	Location	Status
Financial Manipulation	Minor	buyback/DividendDistributor.sol (v1): 100~102	Acknowledged

Description

A sandwich attack may happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (executing before the target) a transaction to purchase one of the assets and make profits by backrunning (executing after the target) a transaction to sell the asset.

```
100         router.swapExactETHForTokensSupportingFeeOnTransferTokens{
101             value: msg.value
102         }(0, path, address(this), block.timestamp);
```

Function `DividendDistributor.deposit` invokes `IUniswapV2Router02.swapExactETHForTokensSupportingFeeOnTransferTokens` without setting restrictions on slippage or minimum output amount. Transactions triggering these functions are vulnerable to sandwich attacks.

Recommendation

It is recommended to add price and slippage control.

DDT-06 | POTENTIAL DIVIDE BY ZERO

Category	Severity	Location	Status
Logical Issue	Minor	buyback/DividendDistributor.sol (v1): 110	Resolved

Description

Performing division by zero would raise an error and revert the transaction.

```
110            dividendsPerShareAccuracyFactor.mul(amount).div(totalShares)
```

The expression `dividendsPerShareAccuracyFactor.mul(amount).div(totalShares)` may divide by zero. Its divisor has has estimated interval [0, 115792089237316195423570985008687907853269984665640564039457584007913129639935].

Recommendation

It is recommended to either reformulate the divisor expression, or to use conditionals or require statements to rule out the possibility of a divide-by-zero.

Alleviation

The client revised the code and resolved this issue in commit : dd4525ef44885bde7f8a8c9b7f2c901e9d33b4a3.

DDT-07 | UNTRUSTED ROUTER ADDRESS RISK

Category	Severity	Location	Status
Logical Issue	Minor	buyback/DividendDistributor.sol (v1): 53	Resolved

Description

The contract allows the router address to be set arbitrarily during contract deployment. This introduces a potential vulnerability where a malicious router can be set during the contract deployment process. A malicious router could potentially behave in unexpected ways, leading to loss of funds or other vulnerabilities being exploited.

Recommendation

We recommend implementing a whitelist of approved router addresses. This whitelist should include addresses for well-known and audited DEXs. Additionally, we recommend including validation of the `router` input variable to only allow router addresses contained in this whitelist.

Alleviation

Code removed in commit 0283052f33840c387e82151caf2dc02263dfa3e7.

ERR-01 | POTENTIAL CROSS-CHAIN REPLAY ATTACK

Category	Severity	Location	Status
Logical Issue	Minor	ERC2612/ERC2612.sol (v1): 100, 112, 113, 125	Resolved

Description

Signed messages are not properly verified with the current chain ID, thus allowing attackers to perform replay attacks across chains. Hardcoded or cached chain ID values are also vulnerable since a hard fork may occur and change the chain ID in the future.

```
100             verifyEIP712(owner, hashStruct, v, r, s) || verifyPersonalSign(
owner, hashStruct, v, r, s),
```

- Calling `verifyPersonalSign`, which eventually calls `ecrecover`.

```
125         address signer = ecrecover(hash, v, r, s);
```

- Calling `ecrecover` with a hash that does not properly include the chain ID.

```
112         bytes32 hash = keccak256(abi.encodePacked('\x19\x01', DOMAIN_SEPARATOR,
 hashStruct));
```

- Reading a state variable `DOMAIN_SEPARATOR`, which seems to use a cached chain ID value.
- Encoding the cached chain ID value `DOMAIN_SEPARATOR`.

```
113         address signer = ecrecover(hash, v, r, s);
```

- Calling `ecrecover` with a hash that does not properly include the chain ID.

Recommendation

We recommend verifying signed messages against the current chain ID by using `block.chainid` or `chainid()` within the same transaction.

Alleviation

ERC2612.sol was removed from the repo as it was unused.

GLOBAL-03 | THIRD-PARTY DEPENDENCY USAGE

Category	Severity	Location	Status
Design Issue	Minor		Acknowledged

Description

The contract is serving as the underlying entity to interact with one or more third party protocols. The scope of the audit treats third party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third parties can possibly create severe impacts, such as increasing fees of third parties, migrating to new LP pools, etc.

```
    IUniswapV2Router02 public router;
```

- The contract `DividendDistributor` interacts with third party contract with `IUniswapV2Router02` interface via `router`.

```
antibot.onPreTransferCheck(sender, recipient, amount);
```

- The contract `StandardToken` interacts with third party contract with `IKARMAAntiBot` interface via `antibot`.

Recommendation

The auditors understood that the business logic requires interaction with third parties. It is recommended for the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

APPENDIX | Playbunny [EX] - AUDIT

Finding Categories

Categories	Description
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.
Concurrency	Concurrency findings are about issues that cause unexpected or unsafe interleaving of code executions.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Financial Manipulation	Financial Manipulation findings indicate issues in design that may lead to financial losses.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. 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 CertiK's prior written consent in each instance.

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 considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report 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 in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK'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 CertiK is subject to dependencies and 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.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | Securing the Web3 World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.