# GMAC-DEPLOYER
# Security Audit Report

**PREPARED FOR:**

**GEMACH DAO**

**ARCADIA CONTACT INFO**

**Email:** audits@arcadiamgroup.com

**Telegram:** https://t.me/thearcadiagroup

**Revision history**

| Date | Reason | Commit |
|---|---|---|
| 03/12/2024 | Initial Audit Scope | #528e69f0c960703fda62d1e0b20341e0e2e1efa5 |
| | Review Of Remediations | |

# Table of Contents

# Executive Summary

## 1. Introduction and Audit Scope

Gemach engaged Arcadia to perform a security audit of their gmac-deployer smart contracts. Our review of their codebase occurred on the commit hash **#528e69f0c960703fda62d1e0b20341e0e2e1efa5**

### a. Review Team

Jihed Chalghaf - Security Researcher and Engineer

Joel Farris - Project Manager

### b. Coverage

For this audit, we performed research, test coverage, investigation, and review of Gemach's gmac-deployer contracts followed by issue reporting, along with mitigation and remediation instructions as outlined in this report. The following code repositories, files, and/or libraries are considered in scope for the review.

| File | Lines | nLines | nSLOC | Comment Lines | Complex. Score |
|---|---|---|---|---|---|
| src/BaseToken.sol | 131 | 126 | 79 | 28 | 55 |
| src/BaseFactory.sol | 122 | 67 | 43 | 25 | 40 |
| src/ERC404/BaseERC404.sol | 55 | 53 | 41 | 6 | 34 |
| src/TaxHelper.sol | 52 | 46 | 28 | 11 | 11 |
| src/ERC404/ERC404TokenFactory.sol | 54 | 49 | 26 | 15 | 26 |
| src/ERC20/ERC20TokenFactory.sol | 51 | 47 | 25 | 14 | 26 |
| src/ERC20/BaseERC20.sol | 29 | 29 | 23 | 5 | 9 |
| src/BaseToken.sol | 131 | 126 | 79 | 28 | 55 |

| | | | | | |
|---|---|---|---|---|---|
| src/BaseFactory.sol | 122 | 67 | 43 | 25 | 40 |
| src/ERC404/BaseERC404.sol | 55 | 53 | 41 | 6 | 34 |
| src/TaxHelper.sol | 52 | 46 | 28 | 11 | 11 |
| src/ERC404/ERC404TokenFactory.sol | 54 | 49 | 26 | 15 | 26 |

## 2. Audit Summary

### a. Audit Methodology

Arcadia completed this security review using various methods, primarily consisting of dynamic and static analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

The followings are the steps we have performed while auditing the smart contracts:

- Investigating the project and its technical architecture overview through its documentation
- Understanding the overview of the smart contracts, the functions of the contracts, the inheritance, and how the contracts interface with each other thanks to the graph created by Solidity Visual Developer
- Manual smart contract audit:
    - Review the code to find any issue that could be exploited by known attacks listed by Consensys
    - Identifying which existing projects the smart contracts are built upon and what are the known vulnerabilities and remediations to the existing projects
    - Line-by-line manual review of the code to find any algorithmic and arithmetic related vulnerabilities compared to what should be done based on the project's documentation
    - Find any potential code that could be refactored to save gas
    - Run through the unit-tests and test-coverage if exists
- Static Analysis:

- Scanning for vulnerabilities in the smart contracts using Static Code Analysis Software
- Making a static analysis of the smart contracts using Slither

- Additional review: a follow-up review is done when the smart contracts have any new updates. The follow-up is done by reviewing all changes compared to the audited commit revision and its impact on the existing source code and found issues.

### b. Summary

There were **15** issues found, **0** of which were deemed to be 'critical,' and **0** of which were rated as 'high'

| Severity Rating | Number of Original Occurrences | Number of Remaining Occurrences |
|---|---|---|
| CRITICAL | 0 | 0 |
| HIGH | 0 | 0 |
| MEDIUM | 1 | 0 |
| LOW | 5 | 4 |
| INFORMATIONAL | 4 | 4 |
| GAS | 5 | 5 |

# Findings in Manual Audit

## 1. Flawed slippage mechanism may result in significant slippage for traders when swapping BaseToken to WNT

### Issue ID

GMAC-1

### Status

Resolved

### Risk Level

Severity: Medium, likelihood: High

### Code Segment

`BaseToken.swap function:`

```
function swap(
        uint256 _amount,
        uint256 _minOut,
        address _receiver,
        bool _fromToken
    ) external nonReentrant {
        if (_amount == 0) revert InvalidAmount();

        uint256 _tax = 0;
        address[] memory _path = new address[](2);
        if (_fromToken) {
            IERC20(address(this)).safeTransferFrom(
                msg.sender,
                address(this),
                _amount
```

```
            );
            IERC20(address(this)).forceApprove(address(univ2router),
_amount);

            _path[0] = address(this);
            _path[1] = address(wnt);
        } else {
            wnt.safeTransferFrom(msg.sender, address(this), _amount);
            wnt.forceApprove(address(univ2router), _amount);

            _path[0] = address(wnt);
            _path[1] = address(this);

            _tax = (_amount * SWAP_TAX) / PRECISION;
            _amount -= _tax;

            wnt.safeTransfer(treasury, _tax);
        }

        // slither-disable-next-line unused-return
        univ2router.swapExactTokensForTokens(
            _amount, // amountIn
            _minOut, // amountOutMin
            _path, // path
            _fromToken ? address(taxHelper) : _receiver, // to
            block.timestamp // deadline
        );

        if (_fromToken)
            _tax = taxHelper.taxAndTransfer(
                SWAP_TAX,
```

```
                PRECISION,
                address(wnt),
                _receiver,
                treasury
            );

        emit Swap(_tax, _fromToken);
    }
```

TaxHelper.taxAndTransfer function:

```
function taxAndTransfer(
        uint256 _tax,
        uint256 _precision,
        address _token,
        address _tokenReceiver,
        address _taxReceiver
    ) external returns (uint256) {
        uint256 _amount = IERC20(_token).balanceOf(address(this));
        uint256 _taxAmount = (_amount * _tax) / _precision;
        uint256 _amountAfterTax = _amount - _taxAmount;

        emit TaxAndTransfer(
            _amountAfterTax,
            _taxAmount,
            _token,
            _tokenReceiver,
            _taxReceiver
        );

        IERC20(_token).safeTransfer(_taxReceiver, _taxAmount);
```

```
        IERC20(_token).safeTransfer(_tokenReceiver, _amountAfterTax);

        return _taxAmount;
    }
```

### Description

During the exchange of a `BaseToken`, the trader stipulates the minimum acceptable quantity of tokens to be received. Should the actual amount fall short of this threshold, the transaction must revert to safeguard traders against slippage.

For each exchange, there is a `0.25%` tax levied in Wrapped Native Token (`WNT`), which becomes applicable if the quantity of `WNT` in or out for each exchange equals or exceeds `400` in Wei.

In the case of swapping `WNT` for `BaseToken`, the tax is subtracted from the exchange amount, and the remaining amount is then utilized during the `Uniswap v2` exchange invocation. This ensures that the trader receives no less than the amount specified using the `_minOut` parameter.

However, when exchanging `BaseToken` for `WNT`, the tax is deducted from the output amount received from the Uniswap v2 exchange invocation, which is then remitted to the `taxHelper`. Consequently, this could potentially result in the trader receiving a lesser amount than anticipated, depending upon the output amount and the tax value.

### Code Location

```
gmac-deployer/src/BaseToken.sol
gmac-deployer/src/TaxHelper.sol
```

### Proof of Concept

Consider adding the below fuzz test into your existing tests at `test/ERC20.t.sol`.

It will fail if the slippage protection for `BaseToken` to `WNT` swaps is broken without the transaction being reverted.

```
function testFuzz_swapSlippage(
        uint256 swapAmount1,
        uint256 swapAmount2,
        uint256 wntSupply,
        uint256 tokenSupply
    ) external {
        vm.selectFork(forkIDs.mainnet);

        vm.assume(swapAmount2 != 0);
        tokenSupply = bound(tokenSupply, 1 ether, 1e26);
        wntSupply = bound(wntSupply, 1 ether, 100 ether);
        swapAmount1 = bound(swapAmount1, 1, wntSupply);

        ERC20TokenFactory _factory = erc20TokenFactoryMainnet;
        IERC20 _wnt = WETH_ETH;
        address _user = userEthereum;

        vm.startPrank(_user);

        _wnt.forceApprove(address(_factory), wntSupply);

        string memory _name = "TestToken";
        string memory _symbol = "TT";
        (address _pair, address _token) = _factory.createERC20{
            value: wntSupply
```

```
        }(_name, _symbol, tokenSupply);

        // Swapping
        BaseToken _baseToken = BaseToken(_token);

        // ************************
        // Swap WNT for Token
        // ************************

        uint256 _userTokenBalanceBefore =
IERC20(address(_baseToken)).balanceOf(
            _user
        );
        _wnt.forceApprove(address(_baseToken), swapAmount1);

        uint256 tax1 = (swapAmount1 * 25) / 10000;
        //@note reverts with Uniswap::INSUFFICIENT_AMOUNT_OUT if tax
is not deducted from the swap amount
        //which is the expected behavior since that's the amount being
sent to univ2router.swapExactTokensForTokens inside the swap()
function
        uint256 minOut1 = UNIV2_ROUTER_ETH.getAmountOut(
            swapAmount1 - tax1,
            wntSupply,
            tokenSupply
        );
        vm.assume(minOut1 ≠ 0);
        _baseToken.swap(swapAmount1, minOut1, _user, false);

        assertTrue(
            IERC20(address(_baseToken)).balanceOf(_user) -
```

```
                _userTokenBalanceBefore >=
                minOut1,
            "Excessive Slippage: WNT in; Token out"
        );

        // ************************
        // Swap Token for WNT
        // ************************

        uint256 _userWntBalanceBefore = _wnt.balanceOf(_user);
        vm.assume(swapAmount2 <=
IERC20(address(_baseToken)).balanceOf(_user));
        IERC20(address(_baseToken)).forceApprove(
            address(_baseToken),
            swapAmount2
        );

        //@note for Token -> WNT swaps, the slippage is being checked
on uniswap before deducting tax
        //which could result in the transaction passing the uniswap
check and sending the trader less WNT than
        //the minimum amount that he has specified, causing an
excessive slippage
        uint256 minOut2 = UNIV2_ROUTER_ETH.getAmountOut(
            swapAmount2,
            tokenSupply,
            wntSupply
        );
        vm.assume(minOut2 != 0);
        _baseToken.swap(swapAmount2, minOut2, _user, true);
```

```
        uint256 _wntEarned = _wnt.balanceOf(_user) -
_userWntBalanceBefore;

        assertTrue(
            _wntEarned ⩾ minOut2,
            "Excessive Slippage: Token in; WNT out"
        );

        vm.stopPrank();
    }
```

### Recommendation

Consider evaluating the slippage parameter as the last step before transferring tokens to users. This can be done by passing the `_minOut` parameter to the `TaxHelper.taxAndTransfer` function and performing the following check before the token transfers.

```
if (_amountAfterTax < _minOut) revert Slippage();
```

## 2. Giving approval to **univ2router** before applying tax may result in a higher than required approval amount

### Issue ID

GMAC-2

### Status

Resolved

### Risk Level

Severity: Low, likelihood: Medium

### Code Segment

```
function swap(
        uint256 _amount,
        uint256 _minOut,
        address _receiver,
        bool _fromToken
    ) external nonReentrant {
        if (_amount == 0) revert InvalidAmount();

        uint256 _tax = 0;
        address[] memory _path = new address[](2);
        if (_fromToken) {
            IERC20(address(this)).safeTransferFrom(
                msg.sender,
                address(this),
                _amount
            );
            IERC20(address(this)).forceApprove(address(univ2router),
_amount);

            _path[0] = address(this);
            _path[1] = address(wnt);
        } else {
            wnt.safeTransferFrom(msg.sender, address(this), _amount);
            wnt.forceApprove(address(univ2router), _amount);

            _path[0] = address(wnt);
            _path[1] = address(this);
```

```
        _tax = (_amount * SWAP_TAX) / PRECISION;
        _amount -= _tax;

        wnt.safeTransfer(treasury, _tax);
    }

    // slither-disable-next-line unused-return
    univ2router.swapExactTokensForTokens(
        _amount, // amountIn
        _minOut, // amountOutMin
        _path, // path
        _fromToken ? address(taxHelper) : _receiver, // to
        block.timestamp // deadline
    );

    if (_fromToken)
        _tax = taxHelper.taxAndTransfer(
            SWAP_TAX,
            PRECISION,
            address(wnt),
            _receiver,
            treasury
        );

    emit Swap(_tax, _fromToken);
}
```

### Description

When swapping from `WNT` to `BaseToken (BaseERC20, BaseERC404)`, the contract grants approval for an amount equivalent to `_amount` to the Uniswap v2 Router. Nonetheless, if the tax value exceeds zero, the actual amount requiring approval becomes the newly adjusted `_amount`, which is smaller. Consequently, this leads to providing an unnecessarily higher approval amount.

### Code Location

`gmac-deployer/src/BaseToken.sol`

### Recommendation

Consider approving after decrementing `_amount` by the `tax` amount.

```
else {
            wnt.safeTransferFrom(msg.sender, address(this), _amount);

            _path[0] = address(wnt);
            _path[1] = address(this);

            _tax = (_amount * SWAP_TAX) / PRECISION;
            _amount -= _tax;

            wnt.forceApprove(address(univ2router), _amount);
            wnt.safeTransfer(treasury, _tax);
        }
```

## 3. Bytecode might not be compatible with all EVM-based chains

### Issue ID

GMAC-3

### Status

Unresolved

### Risk Level

Severity: Low, likelihood: Medium

### Description

The protocol is expected to be deployed on multiple EVM-based chains (Arbitrum, Avalanche, Base, Fraxtal) but the pragma statement shows usage of **0.8.23** version of the Solidity compiler.

This version (and every version after 0.8.19) will use the `PUSH0` opcode, which is still not supported on some EVM-based chains, for example Fraxtal.

### Code Location

```
gmac-deployer/src/*.sol
```

### Recommendation

Consider either using the 0.8.19 version or changing the EVM version to `paris` instead of `shanghai` to avoid `push0` opcodes.

## 4. Using an older version of **OpenZeppelin** libraries may be dangerous

### Issue ID

GMAC-4

### Status

Unresolved

### Risk Level

Severity: Low, likelihood: Low

### Code Segment

```
{
  "name": "openzeppelin-solidity",
  "description": "Secure Smart Contract library for Solidity",
  "version": "5.0.1",
  …
}
```

### Description

Currently the protocol is using version 5.0.1 of the `OpenZeppelin` contracts which are continuously updated to eliminate any bugs and vulnerabilities.

### Code Location

```
gmac-deployer/lib/openzeppelin-contracts/package.json
```

### Recommendation

Consider using the latest version **(5.0.2** so far**)**.

## **5.** Non-conformance to Solidity naming conventions

### Issue ID

GMAC-5

### Status

Unresolved

### Risk Level

Severity: Informational

### Description

Throughout the code base, several non-conformances to Solidity naming convention were spotted. It is advisable to adhere to these naming conventions to improve clarity and avoid ambiguity. These issues are reported through Slither as follows:

```
INFO:Detectors:
Parameter BaseToken.swap(uint256,uint256,address,bool)._amount
(src/BaseToken.sol#72) is not in mixedCase
Parameter BaseToken.swap(uint256,uint256,address,bool)._minOut
(src/BaseToken.sol#73) is not in mixedCase
Parameter BaseToken.swap(uint256,uint256,address,bool)._receiver
(src/BaseToken.sol#74) is not in mixedCase
Parameter BaseToken.swap(uint256,uint256,address,bool)._fromToken
(src/BaseToken.sol#75) is not in mixedCase
Parameter ERC20TokenFactory.createERC20(string,string,uint256)._name
(src/ERC20/ERC20TokenFactory.sol#33) is not in mixedCase
Parameter ERC20TokenFactory.createERC20(string,string,uint256)._symbol
(src/ERC20/ERC20TokenFactory.sol#34) is not in mixedCase
Parameter ERC20TokenFactory.createERC20(string,string,uint256)._totalSupply
(src/ERC20/ERC20TokenFactory.sol#35) is not in mixedCase
Parameter BaseERC404.tokenURI(uint256)._tokenId
(src/ERC404/BaseERC404.sol#48) is not in mixedCase
Parameter ERC404TokenFactory.createERC404(string,string,string,uint96)._name
(src/ERC404/ERC404TokenFactory.sol#34) is not in mixedCase
Parameter
ERC404TokenFactory.createERC404(string,string,string,uint96)._symbol
(src/ERC404/ERC404TokenFactory.sol#35) is not in mixedCase
Parameter
ERC404TokenFactory.createERC404(string,string,string,uint96)._baseURI
(src/ERC404/ERC404TokenFactory.sol#36) is not in mixedCase
Parameter
ERC404TokenFactory.createERC404(string,string,string,uint96)._totalSupply
(src/ERC404/ERC404TokenFactory.sol#37) is not in mixedCase
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._tax
(src/TaxHelper.sol#29) is not in mixedCase
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._precision
(src/TaxHelper.sol#30) is not in mixedCase
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._token
(src/TaxHelper.sol#31) is not in mixedCase
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._tokenRecei
ver (src/TaxHelper.sol#32) is not in mixedCase
```

```
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._taxReceive
r (src/TaxHelper.sol#33) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-
solidity-naming-conventions
```

### Code Location

```
gmac-deployer/src/BaseToken.sol
gmac-deployer/src/ERC20/ERC20TokenFactory.sol
gmac-deployer/src/ERC404/BaseERC404.sol
gmac-deployer/src/ERC404/ERC404TokenFactory.sol
gmac-deployer/src/TaxHelper.sol
```

### Recommendation

Follow the Solidity naming convention.

## 6. Missing or Incomplete **NatSpec**

### Issue ID

GMAC-6

### Status

Unresolved

### Risk Level

Severity: Informational

### Description

The `NatSpec` documentation within the contracts was discovered to be absent or incomplete on the following occasions:

```
src/BaseFactory.sol:99
BaseFactory:InvalidAmount
Natspec is missing
```

```
src/BaseFactory.sol:100
BaseFactory:InvalidAddress
Natspec is missing

src/BaseFactory.sol:92
BaseFactory:AddLiquidityAndBurn
Natspec is missing

src/BaseFactory.sol:93
BaseFactory:TokenCreated
Natspec is missing

src/BaseFactory.sol:58
BaseFactory:_addLiquidityAndBurn
Natspec is missing

src/BaseToken.sol:108
BaseToken:InvalidAmount
Natspec is missing

src/BaseToken.sol:102
BaseToken:Swap
Natspec is missing

src/ERC404/BaseERC404.sol:40
BaseERC404:name
@inheritdoc is missing

src/ERC404/BaseERC404.sol:44
BaseERC404:symbol
```

```
@inheritdoc is missing

src/ERC404/BaseERC404.sol:48
BaseERC404:tokenURI
@inheritdoc is missing

src/ERC404/BaseERC404.sol:15
BaseERC404:_name
Natspec is missing

src/ERC404/BaseERC404.sol:16
BaseERC404:_symbol
Natspec is missing

src/ERC404/BaseERC404.sol:17
BaseERC404:_baseURI
Natspec is missing

src/TaxHelper.sol:12
TaxHelper:TaxAndTransfer
Natspec is missing
```

### Code Location

```
gmac-deployer/src/BaseFactory.sol
gmac-deployer/src/BaseToken.sol
gmac-deployer/src/ERC404/BaseERC404.sol
gmac-deployer/src/TaxHelper.sol
```

### Recommendation

Consider adding the missing **NatSpec** comments and parameters.

## 7. Non-conformance to Solidity order of layout

**Issue ID**

GMAC-7

**Status**

Unresolved

**Risk Level**

Severity: Informational

**Description**

Ordering helps readers identify which functions they can call and to find the constructor and fallback definitions easier.

**Code Location**

```
gmac-deployer/src/BaseFactory.sol
gmac-deployer/src/BaseToken.sol
```

**Recommendation**

Consider applying the order of functions for each contract.

## 8. Use scientific notation for number literals

**Issue ID**

GMAC-8

**Status**

Unresolved

**Risk Level**

Severity: Informational

### Description

The `BaseToken.PRECISION` value could be hard to read:

```
uint256 public constant PRECISION = 10000;
```

### Code Location

```
gmac-deployer/src/BaseToken.sol
```

### Recommendation

Consider changing to:

```
uint256 public constant PRECISION = 1e4;
```

## 9. Explicit initialization with zero is not required for uint

### Issue ID

GMAC-9

### Status

Unresolved

### Risk Level

Severity: Gas

### Code Segment

```
function swap(
        uint256 _amount,
        uint256 _minOut,
```

```
        address _receiver,
        bool _fromToken
    ) external nonReentrant {
        if (_amount == 0) revert InvalidAmount();

        uint256 _tax = 0;
```

```
function _addLiquidityAndBurn(
        address _token
    ) internal returns (address _pair) {
        if (msg.value == 0) revert InvalidAmount();

        payable(address(wnt)).functionCallWithValue(
            abi.encodeWithSignature("deposit()"),
            msg.value
        );

        uint256 _amountToken =
IERC20(_token).balanceOf(address(this));
        uint256 _amountWNT = wnt.balanceOf(address(this));
        // slither-disable-next-line incorrect-equality
        if (_amountToken == 0 || _amountWNT == 0) revert
InvalidAmount();

        _pair = univ2factory.createPair(_token, address(wnt));

        IERC20(_token).forceApprove(address(univ2router),
_amountToken);
        wnt.forceApprove(address(univ2router), _amountWNT);
```

```
uint256 _liquidity = 0;
```

### Description

Explicit initialization with zero is not required for the variables `_tax` and `_liquidity` because uints are zero by default. Removing this will reduce the contract size and save a bit of gas.

### Code Location

```
gmac-deployer/src/BaseToken.sol
gmac-deployer/src/BaseFactory.sol
```

### Recommendation

Consider changing to:

```
uint256 _tax;

uint256 _liquidity;
```

## 10. Check if amounts are greater than zero before performing safe transfers

### Issue ID

GMAC-10

### Status

Unresolved

### Risk Level

Severity: Gas

Code Segment

```
function taxAndTransfer(
        uint256 _tax,
        uint256 _precision,
        address _token,
        address _tokenReceiver,
        address _taxReceiver
    ) external returns (uint256) {
        uint256 _amount = IERC20(_token).balanceOf(address(this));
        uint256 _taxAmount = (_amount * _tax) / _precision;
        uint256 _amountAfterTax = _amount - _taxAmount;

        emit TaxAndTransfer(
            _amountAfterTax,
            _taxAmount,
            _token,
            _tokenReceiver,
            _taxReceiver
        );

        IERC20(_token).safeTransfer(_taxReceiver, _taxAmount);
        IERC20(_token).safeTransfer(_tokenReceiver, _amountAfterTax);

        return _taxAmount;
    }
```

```
function swap(
        uint256 _amount,
```

```
        uint256 _minOut,
        address _receiver,
        bool _fromToken
    ) external nonReentrant {
        if (_amount == 0) revert InvalidAmount();

        uint256 _tax = 0;
        address[] memory _path = new address[](2);
        if (_fromToken) {
            IERC20(address(this)).safeTransferFrom(
                msg.sender,
                address(this),
                _amount
            );
            IERC20(address(this)).forceApprove(address(univ2router),
_amount);

            _path[0] = address(this);
            _path[1] = address(wnt);
        } else {
            wnt.safeTransferFrom(msg.sender, address(this), _amount);
            wnt.forceApprove(address(univ2router), _amount);

            _path[0] = address(wnt);
            _path[1] = address(this);

            _tax = (_amount * SWAP_TAX) / PRECISION;
            _amount -= _tax;

            wnt.safeTransfer(treasury, _tax);
        }
```

### Description

Invoking `safeTransfer` when the amount is zero results in gas wastage, as no funds will be transferred. Additionally, decrementing `_amount` when the tax value equals zero results in unnecessary gas usage.

### Code Location

```
gmac-deployer/src/BaseToken.sol
gmac-deployer/src/TaxHelper.sol
```

### Recommendation

Verify whether the tax amount exceeds zero before decrementing `_amount` and executing the `safeTransfer` function.

Consider implementing the below changes in the `BaseToken.swap` function:

```
_tax = (_amount * SWAP_TAX) / PRECISION;

if (_tax ≠ 0) {
    _amount -= _tax;
    wnt.safeTransfer(treasury, _tax);
}
```

Consider implementing the below changes in the `TaxHelper.taxAndTransfer` function:

```
uint256 _amount = IERC20(_token).balanceOf(address(this));
uint256 _taxAmount = (_amount * _tax) / _precision;

if (_taxAmount ≠ 0) {
    _amount -= _taxAmount;
```

```
    IERC20(_token).safeTransfer(_taxReceiver, _taxAmount);
}

emit TaxAndTransfer(
    _amount,
    _taxAmount,
    _token,
    _tokenReceiver,
    _taxReceiver
);

IERC20(_token).safeTransfer(_tokenReceiver, _amount);
```

## 11. Using **memory** instead of **calldata** results in more gas

### Issue ID

GMAC-11

### Status

Unresolved

### Risk Level

Severity: Gas

### Code Segment

```
function createERC404(
        string memory _name,
        string memory _symbol,
        string memory _baseURI,
        uint96 _totalSupply
    ) external payable nonReentrant returns (address, address)
```

```
function createERC20(
        string memory _name,
        string memory _symbol,
        uint256 _totalSupply
    ) external payable nonReentrant returns (address, address) {
```

### Description

Loading function inputs or data directly from `calldata` is cheaper than loading from `memory`. This is because accessing data from `calldata` requires fewer operations and gas costs. As a result, it is recommended to use `memory` only when data needs to be updated in the function.

### Code Location

```
gmac-deployer/src/ERC20/ERC20TokenFactory.sol
gmac-deployer/src/ERC404/ERC404TokenFactory.sol
```

### Recommendation

Consider using `calldata` rather than `memory` in the aforementioned appearances.

## 12. Split revert statements

### Issue ID

GMAC-12

### Status

Unresolved

### Risk Level

Severity: Gas

### Code Segment

```
function _addLiquidityAndBurn(
        address _token
    ) internal returns (address _pair) {
        if (msg.value == 0) revert InvalidAmount();

        payable(address(wnt)).functionCallWithValue(
            abi.encodeWithSignature("deposit()"),
            msg.value
        );

        uint256 _amountToken =
IERC20(_token).balanceOf(address(this));
        uint256 _amountWNT = wnt.balanceOf(address(this));
        // slither-disable-next-line incorrect-equality
        if (_amountToken == 0 || _amountWNT == 0) revert
InvalidAmount();
```

### Description

When splitting revert statements, we assert that the validity of each statement is imperative for the function's ongoing execution.

If the initial statement proves false, the function will promptly revert, and subsequent require statements will not undergo evaluation. This measure aims to economize gas usage by bypassing the evaluation of subsequent require statements.

### Code Location

```
gmac-deployer/src/BaseFactory.sol
```

### Recommendation

Consider changing to:

```
if (_amountToken == 0) revert InvalidAmount();
if (_amountWNT == 0) revert InvalidAmount();
```

## 13. Use named returns to save gas

### Issue ID

GMAC-13

### Status

Unresolved

### Risk Level

Severity: Gas

### Code Segment

```
function taxAndTransfer(
        uint256 _tax,
        uint256 _precision,
        address _token,
        address _tokenReceiver,
        address _taxReceiver
    ) external returns (uint256) {
        uint256 _amount = IERC20(_token).balanceOf(address(this));
        uint256 _taxAmount = (_amount * _tax) / _precision;
```

```
        uint256 _amountAfterTax = _amount - _taxAmount;

        emit TaxAndTransfer(
            _amountAfterTax,
            _taxAmount,
            _token,
            _tokenReceiver,
            _taxReceiver
        );

        IERC20(_token).safeTransfer(_taxReceiver, _taxAmount);
        IERC20(_token).safeTransfer(_tokenReceiver, _amountAfterTax);

        return _taxAmount;
    }
```

```
function createERC20(
        string memory _name,
        string memory _symbol,
        uint256 _totalSupply
    ) external payable nonReentrant returns (address, address) {
        BaseERC20 _token = new BaseERC20(
            wnt,
            univ2router,
            taxHelper,
            treasury,
            _name,
            _symbol,
            _totalSupply
```

```
        );

        emit TokenCreated(address(_token), _name, _symbol,
_totalSupply);

        return (_addLiquidityAndBurn(address(_token)),
address(_token));
    }
```

```
function createERC404(
        string memory _name,
        string memory _symbol,
        string memory _baseURI,
        uint96 _totalSupply
    ) external payable nonReentrant returns (address, address) {
        BaseERC404 _token = new BaseERC404(
            wnt,
            univ2router,
            taxHelper,
            treasury,
            _name,
            _symbol,
            _baseURI,
            _totalSupply
        );

        emit TokenCreated(address(_token), _name, _symbol,
_totalSupply);
```

```
        return (_addLiquidityAndBurn(address(_token)),
address(_token));
    }
```

### Description

The solidity compiler outputs more efficient code when the variable is declared in the return statement.

### Code Location

```
gmac-deployer/src/TaxHelper.sol
gmac-deployer/src/ERC20/ERC20TokenFactory.sol
gmac-deployer/src/ERC404/ERC404TokenFactory.sol
```

### Recommendation

Consider changing to:

```
/// @return _taxAmount The tax amount
    function taxAndTransfer(
        uint256 _tax,
        uint256 _precision,
        address _token,
        address _tokenReceiver,
        address _taxReceiver
    ) external returns (uint256 _taxAmount) {
        uint256 _amount = IERC20(_token).balanceOf(address(this));
        _taxAmount = (_amount * _tax) / _precision;
        uint256 _amountAfterTax = _amount - _taxAmount;

        emit TaxAndTransfer(
```

```
            _amountAfterTax,
            _taxAmount,
            _token,
            _tokenReceiver,
            _taxReceiver
        );

        IERC20(_token).safeTransfer(_taxReceiver, _taxAmount);
        IERC20(_token).safeTransfer(_tokenReceiver, _amountAfterTax);
    }
```

```
/// @return _pair The pair address
/// @return _token The token address
    function createERC20(
        string memory _name,
        string memory _symbol,
        uint256 _totalSupply
    ) external payable nonReentrant returns (address _pair, address
_token) {
        _token = address(
            new BaseERC20(
                wnt,
                univ2router,
                taxHelper,
                treasury,
                _name,
                _symbol,
                _totalSupply
            )
        );
```

```
        emit TokenCreated(_token, _name, _symbol, _totalSupply);

        _pair = _addLiquidityAndBurn(_token);
    }
```

```
/// @return _pair The pair address
/// @return _token The token address
    function createERC404(
        string memory _name,
        string memory _symbol,
        string memory _baseURI,
        uint96 _totalSupply
    ) external payable nonReentrant returns (address _pair, address
_token) {
        _token = address(
            new BaseERC404(
                wnt,
                univ2router,
                taxHelper,
                treasury,
                _name,
                _symbol,
                _baseURI,
                _totalSupply
            )
        );

        emit TokenCreated(_token, _name, _symbol, _totalSupply);
```

```
        _pair = _addLiquidityAndBurn(_token);
    }
```

# Automated Audit

## Static Analysis with Slither

We run a static analysis against the source code using Slither, which is a Solidity static analysis framework written in Python 3. Slither runs a suite of vulnerability detectors, prints visual information about contract details. Slither enables developers to find vulnerabilities, enhance their code comprehension, and quickly prototype custom analyses.

The following shows the results found by the static analysis by Slither. We reviewed the results, and, except the issues that were identified previously, all the other issues found by Slither are false positives.

```
 INFO:Detectors:
BaseERC404.tokenURI(uint256) (src/ERC404/BaseERC404.sol#48-50) calls
abi.encodePacked() with multiple dynamic arguments:        - _result =
string(abi.encodePacked(_baseURI,Strings.toString(_tokenId)))
(src/ERC404/BaseERC404.sol#49)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#abi-encodePacked-collis
ion
INFO:Detectors:
BaseToken.constructor(IERC20,IUniswapV2Router01,TaxHelper,address)
(src/BaseToken.sol#31-44) ignores return value by
_delegationRegistry.call(abi.encodeWithSignature(setDelegationForSelf(address),_treas
ury)) (src/BaseToken.sol#41)
BaseToken.constructor(IERC20,IUniswapV2Router01,TaxHelper,address)
(src/BaseToken.sol#49) ignores return value by
_delegationRegistry.call(abi.encodeWithSignature(disableSelfManagingDelegations()))
(src/BaseToken.sol#42)
BaseFactory.constructor(IERC20,IUniswapV2Router01,IUniswapV2Factory,TaxHelper,address
) (src/BaseFactory.sol#30-52) ignores return value by
_delegationRegistry.call(abi.encodeWithSignature(setDelegationForSelf(address),_treas
ury)) (src/BaseFactory.sol#49)
BaseFactory.constructor(IERC20,IUniswapV2Router01,IUniswapV2Factory,TaxHelper,address
) (src/BaseFactory.sol#30-52) ignores return value by
_delegationRegistry.call(abi.encodeWithSignature(disableSelfManagingDelegations()))
(src/BaseFactory.sol#50)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#unchecked-low-level-cal
ls
INFO:Detectors:
BaseFactory._addLiquidityAndBurn(address) (src/BaseFactory.sol#58-86) ignores return
value by
address(address(wnt)).functionCallWithValue(abi.encodeWithSignature(deposit()),msg.va
lue) (src/BaseFactory.sol#61)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
```

```
Reentrancy in BaseFactory._addLiquidityAndBurn(address) (src/BaseFactory.sol#58-86):
        External calls:
        -
address(address(wnt)).functionCallWithValue(abi.encodeWithSignature(deposit()),msg.va
lue) (src/BaseFactory.sol#61)
        - _pair = univ2factory.createPair(_token,address(wnt))
(src/BaseFactory.sol#68)
        - IERC20(_token).forceApprove(address(univ2router),_amountToken)
(src/BaseFactory.sol#70)
        - wnt.forceApprove(address(univ2router),_amountWNT) (src/BaseFactory.sol#71)
        - (_amountToken,_amountWNT,_liquidity) =
univ2router.addLiquidity(_token,address(wnt),_amountToken,_amountWNT,_amountToken,_am
ountWNT,address(0),block.timestamp) (src/BaseFactory.sol#74-83)
        Event emitted after the call(s):
        - AddLiquidityAndBurn(_amountToken,_amountWNT,_liquidity,_pair)
(src/BaseFactory.sol#85)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilit
ies-3
INFO:Detectors:
BaseFactory._addLiquidityAndBurn(address) (src/BaseFactory.sol#58-86) uses timestamp
for comparisons
        Dangerous comparisons:
        - _amountToken == 0 || _amountWNT == 0 (src/BaseFactory.sol#66)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Pragma version0.8.23 (src/BaseFactory.sol#2) necessitates a version too recent to be
trusted. Consider deploying with 0.8.18.
Pragma version0.8.23 (src/BaseToken.sol#2) necessitates a version too recent to be
trusted. Consider deploying with 0.8.18.
Pragma version0.8.23 (src/ERC20/BaseERC20.sol#2) necessitates a version too recent to
be trusted. Consider deploying with 0.8.18.
Pragma version0.8.23 (src/ERC20/ERC20TokenFactory.sol#2) necessitates a version too
recent to be trusted. Consider deploying with 0.8.18.
Pragma version0.8.23 (src/ERC404/BaseERC404.sol#2) necessitates a version too recent
to be trusted. Consider deploying with 0.8.18.
Pragma version0.8.23 (src/ERC404/ERC404TokenFactory.sol#2) necessitates a version too
recent to be trusted. Consider deploying with 0.8.18.
Pragma version0.8.23 (src/TaxHelper.sol#2) necessitates a version too recent to be
trusted. Consider deploying with 0.8.18.
solc-0.8.23 is not recommended for deployment
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-s
olidity
INFO:Detectors:
Low level call in
BaseFactory.constructor(IERC20,IUniswapV2Router01,IUniswapV2Factory,TaxHelper,address
) (src/BaseFactory.sol#30-52):
        -
_delegationRegistry.call(abi.encodeWithSignature(setDelegationForSelf(address),_treas
ury)) (src/BaseFactory.sol#49)
        -
_delegationRegistry.call(abi.encodeWithSignature(disableSelfManagingDelegations()))
(src/BaseFactory.sol#50)
Low level call in BaseToken.constructor(IERC20,IUniswapV2Router01,TaxHelper,address)
(src/BaseToken.sol#31-44):
```

```
        -
_delegationRegistry.call(abi.encodeWithSignature(setDelegationForSelf(address),_treas
ury)) (src/BaseToken.sol#41)
        -
_delegationRegistry.call(abi.encodeWithSignature(disableSelfManagingDelegations()))
(src/BaseToken.sol#42)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter BaseToken.swap(uint256,uint256,address,bool)._amount (src/BaseToken.sol#56)
is not in mixedCase
Parameter BaseToken.swap(uint256,uint256,address,bool)._minOut (src/BaseToken.sol#57)
is not in mixedCase
Parameter BaseToken.swap(uint256,uint256,address,bool)._receiver
(src/BaseToken.sol#58) is not in mixedCase
Parameter BaseToken.swap(uint256,uint256,address,bool)._fromToken
(src/BaseToken.sol#59) is not in mixedCase
Parameter ERC20TokenFactory.createERC20(string,string,uint256)._name
(src/ERC20/ERC20TokenFactory.sol#34) is not in mixedCase
Parameter ERC20TokenFactory.createERC20(string,string,uint256)._symbol
(src/ERC20/ERC20TokenFactory.sol#35) is not in mixedCase
Parameter ERC20TokenFactory.createERC20(string,string,uint256)._totalSupply
(src/ERC20/ERC20TokenFactory.sol#36) is not in mixedCase
Parameter BaseERC404.tokenURI(uint256)._tokenId (src/ERC404/BaseERC404.sol#48) is not
in mixedCase
Parameter ERC404TokenFactory.createERC404(string,string,string,uint96)._name
(src/ERC404/ERC404TokenFactory.sol#35) is not in mixedCase
Parameter ERC404TokenFactory.createERC404(string,string,string,uint96)._symbol
(src/ERC404/ERC404TokenFactory.sol#36) is not in mixedCase
Parameter ERC404TokenFactory.createERC404(string,string,string,uint96)._baseURI
(src/ERC404/ERC404TokenFactory.sol#37) is not in mixedCase
Parameter ERC404TokenFactory.createERC404(string,string,string,uint96)._totalSupply
(src/ERC404/ERC404TokenFactory.sol#38) is not in mixedCase
Parameter TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._tax
(src/TaxHelper.sol#24) is not in mixedCase
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._precision
(src/TaxHelper.sol#25) is not in mixedCase
Parameter TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._token
(src/TaxHelper.sol#26) is not in mixedCase
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._tokenReceiver
(src/TaxHelper.sol#27) is not in mixedCase
Parameter
TaxHelper.taxAndTransfer(uint256,uint256,address,address,address)._taxReceiver
(src/TaxHelper.sol#28) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity
-naming-conventions
INFO:Slither:. analyzed (26 contracts with 93 detectors), 35 result(s) found
```

## Unit Tests

```
➜  gmac-deployer git:(main) ✗ forge test -vvv --match-path test/ERC20.t.sol
[⠊] Compiling...
No files changed, compilation skipped

Running 6 tests for test/ERC20.t.sol:ERC20
[PASS] testArbitrum() (gas: 4053314)
[PASS] testAvalanche() (gas: 4021930)
[PASS] testEthereum() (gas: 4020594)
[PASS] testFraxtal() (gas: 6316188)
[PASS] testGoerli() (gas: 4022457)
[FAIL. Reason: Assertion failed.] testSepolia() (gas: 3630739)
Logs:
  Error: _testSwap: E3
  Error: a == b not satisfied [uint]
        Left: 102500000000000000
       Right: 2500000000000000
```

The `testSepolia()` test fails because the `TREASURY` address has an initial `WNT` balance of 0.1 ether as shown below:

```
function _testSwap(BaseToken _token, IERC20 _wnt, address _user)
internal {
        // ************************
        // Swap WNT for Token
        // ************************

        vm.startPrank(_user);

        uint256 _userWntBalanceBefore = _wnt.balanceOf(_user);
        uint256 _userTokenBalanceBefore =
IERC20(address(_token)).balanceOf(
            _user
        );
        uint256 _amount = 1 ether;
        _wnt.forceApprove(address(_token), _amount);
```

```
        console.log(
            "Initial WNT Balance of TREASURY: ",
            _wnt.balanceOf(TREASURY)
        );
        _token.swap(_amount, 0, _user, false);
```

```
Running 6 tests for test/ERC20.t.sol:ERC20
[PASS] testArbitrum() (gas: 4058265)
Logs:
  Initial WNT Balance of TREASURY:  0

[PASS] testAvalanche() (gas: 4026076)
Logs:
  Initial WNT Balance of TREASURY:  0

[PASS] testEthereum() (gas: 4024783)
Logs:
  Initial WNT Balance of TREASURY:  0

[PASS] testFraxtal() (gas: 6320680)
Logs:
  Initial WNT Balance of TREASURY:  0

[PASS] testGoerli() (gas: 4026664)
Logs:
  Initial WNT Balance of TREASURY:  0

[FAIL. Reason: Assertion failed.] testSepolia() (gas: 3634841)
Logs:
  Initial WNT Balance of TREASURY:  100000000000000000
  Error: _testSwap: E3
  Error: a == b not satisfied [uint]
        Left: 102500000000000000
       Right: 2500000000000000
```

**Issue ID**

GMAC-14

### Risk Level

Severity: Low, Likelihood: Medium

### Description

The test case failed due to an incorrect assumption that the `TREASURY`'s initial `WNT` balance should be zero across all chains.

### Code Location

`gmac-deployer/test/Base.t.sol`

### Recommendation

Consider changing the `_testSwap()` function to the following:

```
function _testSwap(BaseToken _token, IERC20 _wnt, address _user)
internal {
        // ************************
        // Swap WNT for Token
        // ************************

        vm.startPrank(_user);

        uint256 _userWntBalanceBefore = _wnt.balanceOf(_user);
        uint256 _userTokenBalanceBefore =
IERC20(address(_token)).balanceOf(
            _user
        );
        uint256 _treasuryWntBalanceBefore = _wnt.balanceOf(TREASURY);
        uint256 _amount = 1 ether;
        _wnt.forceApprove(address(_token), _amount);
        _token.swap(_amount, 0, _user, false);

        assertTrue(
```

```
            IERC20(address(_token)).balanceOf(_user) > 0,
            "_testSwap: E1"
        );
        assertEq(
            _wnt.balanceOf(_user),
            _userWntBalanceBefore - 1 ether,
            "_testSwap: E2"
        );
        assertEq(
            _wnt.balanceOf(TREASURY),
            _treasuryWntBalanceBefore + 0.0025 ether,
            "_testSwap: E3"
        );
```

```
→  gmac-deployer git:(main) ✗ forge test -vvv --match-path test/ERC20.t.sol
[⠒] Compiling...
No files changed, compilation skipped

Running 6 tests for test/ERC20.t.sol:ERC20
[PASS] testArbitrum() (gas: 4055052)
[PASS] testAvalanche() (gas: 4022863)
[PASS] testEthereum() (gas: 4021570)
[PASS] testFraxtal() (gas: 6317467)
[PASS] testGoerli() (gas: 4023451)
[PASS] testSepolia() (gas: 3616692)
Test result: ok. 6 passed; 0 failed; 0 skipped; finished in 15.50s
Ran 1 test suites: 6 tests passed, 0 failed, 0 skipped (6 total tests)
```

Regarding the `ERC404` tests, they were consuming approximately `12 GB` of RAM and consistently failing.

```
→  gmac-deployer git:(main) ✗ forge test -vvv --match-path test/ERC404.t.sol
[⠒] Compiling...
No files changed, compilation skipped
[1]    8863 killed     forge test -vvv --match-path test/ERC404.t.sol
```

Executing the tests individually led to all of them passing successfully.

```
→  gmac-deployer git:(main) ✗ forge test --match-path test/ERC404.t.sol --match-test testArbitrum
[⠒] Compiling...
No files changed, compilation skipped

Running 1 test for test/ERC404.t.sol:ERC404
[PASS] testArbitrum() (gas: 13266078887)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 26.12s
Ran 1 test suites: 1 tests passed, 0 failed, 0 skipped (1 total tests)
→  gmac-deployer git:(main) ✗ forge test --match-path test/ERC404.t.sol --match-test testEthereum
[⠒] Compiling...
No files changed, compilation skipped

Running 1 test for test/ERC404.t.sol:ERC404
[PASS] testEthereum() (gas: 13266052061)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 26.36s
Ran 1 test suites: 1 tests passed, 0 failed, 0 skipped (1 total tests)
→  gmac-deployer git:(main) ✗ forge test --match-path test/ERC404.t.sol --match-test testFraxtal
[⠢] Compiling...
No files changed, compilation skipped

Running 1 test for test/ERC404.t.sol:ERC404
[PASS] testFraxtal() (gas: 13267952499)
```

```
Running 1 test for test/ERC404.t.sol:ERC404
[PASS] testAvalanche() (gas: 13266053096)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 23.72s
Ran 1 test suites: 1 tests passed, 0 failed, 0 skipped (1 total tests)
→  gmac-deployer git:(main) ✗ forge test --match-path test/ERC404.t.sol --match-test testGoerli
[⠒] Compiling...
No files changed, compilation skipped

Running 1 test for test/ERC404.t.sol:ERC404
[PASS] testGoerli() (gas: 13266053606)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 29.05s
Ran 1 test suites: 1 tests passed, 0 failed, 0 skipped (1 total tests)
→  gmac-deployer git:(main) ✗ forge test --match-path test/ERC404.t.sol --match-test testSepolia
[⠢] Compiling...
No files changed, compilation skipped

Running 1 test for test/ERC404.t.sol:ERC404
[PASS] testSepolia() (gas: 13265730479)
Test result: ok. 1 passed; 0 failed; 0 skipped; finished in 25.17s
Ran 1 test suites: 1 tests passed, 0 failed, 0 skipped (1 total tests)
```

## Tests Coverage

| File | % Lines | % Statements | % Branches | % Funcs |
|---|---|---|---|---|
| src/BaseFactory.sol | 100.00% (11/11) | 88.24% (15/17) | 50.00% (2/4) | 100.00% (1/1) |
| src/BaseToken.sol | 100.00% (19/19) | 94.74% (18/19) | 50.00% (3/6) | 100.00% (1/1) |
| src/ERC20/ERC20TokenFactory.sol | 100.00% (3/3) | 100.00% (4/4) | 100.00% (0/0) | 100.00% (1/1) |
| src/ERC404/BaseERC404.sol | 0.00% (0/4) | 0.00% (0/4) | 0.00% (0/2) | 0.00% (0/3) |
| src/ERC404/ERC404TokenFactory.sol | 100.00% (3/3) | 100.00% (4/4) | 100.00% (0/0) | 100.00% (1/1) |
| src/TaxHelper.sol | 100.00% (7/7) | 100.00% (10/10) | 100.00% (0/0) | 100.00% (1/1) |
| Total | 91.49% (43/47) | 87.93% (51/58) | 41.67% (5/12) | 62.50% (5/8) |

For the `BaseERC404` contract, the uncovered lines are related to not calling the `name()`, `symbol()` and `tokenURI()` functions.

```
39          :     function name() public view override returns (string memory) {
40        0 :         return _name;
41          :     }
42          :
43          :     function symbol() public view override returns (string memory) {
44        0 :         return _symbol;
45          :     }
46          :
47          :     function tokenURI(
48          :         uint256 _tokenId
49          :     ) public view override returns (string memory _result) {
50        0 :         if (bytes(_baseURI).length != 0)
51        0 :             _result = string(
52          :                 abi.encodePacked(_baseURI, Strings.toString(_tokenId))
53          :             );
54          :     }
```

The contract's tests are a bit lacking. It is therefore highly recommended to create suitable test-cases to cover what is missing.

### Issue ID

GMAC-15

### Risk Level

Severity: Low, Likelihood: Low

### Description

Missing unit tests to cover: `BaseERC404.name()`, `BaseERC404.symbol()` and `BaseERC404.tokenURI()`.

### Code Location

```
gmac-deployer/src/ERC404/BaseERC404.sol
```

### Recommendation

Write additional test-cases to cover the uncovered code.


# Disclaimer

While best efforts and precautions have been taken in the preparation of this document, Arcadia and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.