



BASE PROTOCOL

Smart Contract Security Audit

Prepared by: Halborn
Date of Engagement: October 12th, 2020
Visit: Halborn.com

Document Revision History	3
Contacts	3
1 Executive Summary	4
1.1 Introduction	4
1.2 Test Approach and Methodology	5
1.3 SCOPE	5
2 Assessment Summary And Findings Overview	6
3 Findings & Technical Details	7
3.1 Use Of Tx.Origin - Medium	8
Description	8
Code Location	8
Recommendation	8
3.2 Avoid Using Now - Low	9
Description	9
Code Location	9
Recommendation	9
3.3 Balance Validation Is Missing - Low	10
Description	10
Results	10
3.4 State Variable Shadowing - Informational	11
Description	11
Results	12
3.5 Static Analysis - Low	13
Description	13
Results	15
3.6 Solgraph - Informational	16
Description	16
Results	16

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/12/2020	Gabi Urrutia
0.2	Document Edits	10/12/2020	Steven Walbroehl
1.0	Draft Version	10/30/2020	Steven Walbroehl

CONTACTS

CONTACT	COMPANY	EMAIL
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com

1.1 INTRODUCTION

Base Protocol engaged Halborn to conduct a security assessment on their all smart contracts that implement the protocol on the Ethereum blockchain. Base Protocol is a decentralized elastic supply protocol for creating indices of other tokens. It maintains a price peg by adjusting supply directly to and from wallet holders based on a data feed generated by off-chain oracles. The security assessment was scoped to the contract BaseToken, BaseTokenMonetaryPolicy, BaseTokenOrchestrator, ERC20UpgradeSafe, ERC677Token, Greeter and an audit of the security risk and implications regarding the changes introduced by the development team at Base Protocol prior to its production release shortly following the assessments deadline.

Overall, the smart contract code is extremely well documented, follows a high-quality software development standard, contains many utilities and automation scripts to support continuous deployment / testing / integration, and does NOT contain any obvious exploitation vectors that Halborn was able to leverage within the timeframe of testing allotted.

Though the outcome of this security audit is satisfactory; due to time and resource constraints, only testing and verification of essential properties were performed to achieve objectives and deliverables set in the scope. It is important to remark the use of the best practices for secure smart contract development. Halborn recommends performing further testing to validate extended safety and correctness in context to the whole set of contracts. External threats, such as economic attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

1.2 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose
- Smart Contract manual code read and walkthrough
- Graphing out functionality and contract logic/connectivity/functions ([solgraph](#))
- Manual Assessment of use and safety for the critical solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Scanning of solidity files for vulnerabilities, security hotspots, or bugs. ([MythX](#))
- Static Analysis of security for scoped contract, and imported functions. ([Slither](#))
- Testnet deployment ([Truffle](#), [Ganache](#), [Infura](#))
- Smart Contract Fuzzing and dynamic state exploitation ([Echidna](#)) Symbolic Execution / EVM bytecode security assessment ([limited time](#))

1.3 SCOPE

Code related to:

- `BaseToken.sol`
- `BaseTokenMonetaryPolicy.sol`
- `BaseTokenOrchestrator.sol`
- `ERC20UpgradeSafe.sol`
- `ERC677Token.sol`
- `Greeter.sol`

Specific commit of contract: Commit ID:
`cbbf4d8e1970e72df80b461358552c431296c6a2`

OUT-OF-SCOPE:

External contracts, External Oracles, other smart contracts in the repository or imported by BASE protocol contracts, economic attacks

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW
0	0	1	3

SECURITY ANALYSIS	RISK LEVEL
USE OF TX.ORIGIN	Medium
AVOID USING NOW	Low
BALANCE VALIDATION IS MISSING	Low
STATE VARIABLE SHADOWING	Informational
STATIC ANALYSIS	Low



FINDINGS & TECH DETAILS



3.1 USE OF TX.ORIGIN - MEDIUM

Description

"tx.origin" is useful only in very exceptional cases. If it is used for authentication, then it makes no impact, because any contract you call can act on your behalf. So it is recommended to Never use tx.origin for authorization.

Here in `rebase()` function of `BaseTokenOrchestrator.sol` contract which is callable from external has this `require()` condition which should be fixed to filter out non required address to call this method.

Reference:

<https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseTokenOrchestrator.sol#L46>

Code Location

BaseTokenOrchestrator.sol

```

35 *
36 * @notice Main entry point to initiate a rebase operation.
37 * The BaseTokenOrchestrator calls rebase on the policy and notifies downstream applications.
38 * Contracts are guarded from calling, to avoid flash loan attacks on liquidity
39 * providers.
40 * If a transaction in the transaction list reverts, it is swallowed and the remaining
41 * transactions are executed.
42 */
43 function rebase()
44     external
45 {
46     require(msg.sender == tx.origin); // solhint-disable-line avoid-tx-origin
47
48     policy.rebase();
49
50     for (uint i = 0; i < transactionEnabled.length; i++) {
51         // Transaction storage t = transactions[i];
52         if (transactionEnabled[i]) {
53             bool result = externalCall(transactionDestination[i], transactionData[i]);
54             if (!result) {
55                 emit TransactionFailed(transactionDestination[i], i, transactionData[i]);
56                 revert("Transaction Failed");
57             }
58         }
59     }
60 }
61

```

3.2. AVOID USING NOW - LOW

Description:

“now” can be influenced by miners to some degree. Miners can manipulate “now” to exploit the contract. Here in `BaseTokenMonetaryPolicy.sol` contract, “now” is being used in `inRebaseWindow()` and `rebase()` methods.

Recommendation:

Avoid getting relied on use of “now” in require methods.

Code Location:

`BaseTokenMonetaryPolicy.sol`

<https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseTokenMonetaryPolicy.sol#L100>

`BaseTokenMonetaryPolicy.sol`

<https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseTokenMonetaryPolicy.sol#L103>

`BaseTokenMonetaryPolicy.sol`

<https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseTokenMonetaryPolicy.sol#L134>

```

94  */
95  function rebase() external {
96      require(msg.sender == orchestrator, "you are not the orchestrator");
97      require(inRebaseWindow(), "the rebase window is closed");
98
99      // This comparison also ensures there is no reentrancy.
100     require(lastRebaseTimestampSec.add(minRebaseTimeIntervalSec) < now, "cannot rebase yet");
101
102     // Snap the rebase time to the start of this window.
103     lastRebaseTimestampSec = now.sub(now.mod(minRebaseTimeIntervalSec)).add(rebaseWindowOffsetSec);
104
105     epoch = epoch.add(1);
106

```

`BaseTokenMonetaryPolicy.sol`

<https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/Ba>

[seTokenMonetaryPolicy.sol#L257](#)

BaseTokenMonetaryPolicy.sol

[https://github.com/Base-Protocol/contracts-](https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseTokenMonetaryPolicy.sol#L258)

[0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/Ba](https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseTokenMonetaryPolicy.sol#L258)

[seTokenMonetaryPolicy.sol#L258](#)

```

253      *           Otherwise, returns false.
254      */
255      function inRebaseWindow() public view returns (bool) {
256          return (
257              now.mod(minRebaseTimeIntervalSec) >= rebaseWindowOffsetSec &&
258              now.mod(minRebaseTimeIntervalSec) < (rebaseWindowOffsetSec.add(rebaseWindowLengthSec))
259          );
260      }
261  
```

3.3 BALANCE VALIDATION IS MISSING - LOW

Description:

In transferFrom() method of BaseToken.sol, before subtracting the amount from the _shareBalances[], check is missing whether the sufficient amount is there in or not. It will result into integer overflow issue.

Location:

BaseToken.sol

[https://github.com/Base-Protocol/contracts-](https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseToken.sol#L233)

[0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/Bas](https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseToken.sol#L233)

[eToken.sol#L233](#)

```

227 ▾ /**
228      * @dev Transfer tokens from one address to another.
229      * @param from The address you want to send tokens from.
230      * @param to The address you want to transfer to.
231      * @param value The amount of tokens to be transferred.
232      */
233      function transferFrom(address from, address to, uint256 value)
234      public
235      override
236      validRecipient(to)
237      returns (bool)
238      {
239          require(bannedUsers[msg.sender] == false, "you are banned");
240
241          _allowedBASE[from][msg.sender] = _allowedBASE[from][msg.sender].sub(value);
242
243          uint256 shareValue = value.mul(_sharesPerBASE);
244          _shareBalances[from] = _shareBalances[from].sub(shareValue);
245          _shareBalances[to] = _shareBalances[to].add(shareValue);
246          emit Transfer(from, to, value);
247
248          return true;
249      }
250

```

Here, before subtracting the amount from `_shareBalances` of that address, the sufficient amount check is required.

3.4 STATE VARIABLE SHADOWING - INFORMATIONAL

Description:

There are few state variables are getting shadowed by the child contract `BaseToken.sol` (`_totalSupply` variable) and `ERC20UpgradeSafe.sol` (`__gap` variable).

Location:

`_totalSupply` variable from `BaseToken.sol` is shadowing

`_totalSupply` variable of `ERC20UpgradeSafe.sol`

<https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/BaseToken.sol#L144>

```

132
133     function initialize()
134     public
135     initializer
136     {
137         __ERC20_init("Base Protocol", "BASE");
138         _setupDecimals(uint8(DECIMALS));
139         __Ownable_init();
140
141         rebasePausedDeprecated = false;
142         tokenPausedDeprecated = false;
143
144         _totalSupply = INITIAL_SHARES_SUPPLY;
145         _shareBalances[owner()] = TOTAL_SHARES;
146         _sharesPerBASE = TOTAL_SHARES.div(_totalSupply);
147
148         // Ban the Kucoin hacker
149         bannedUsers[0xeB31973E0FeBF3e3D7058234a5e8bAe1aB488c23] = true;
150
151         emit Transfer(address(0x0), owner(), _totalSupply);
152     }
153

```

1 BaseToken.sol

```

31 // allowances: see __ERC20_approve()
32 */
33 contract ERC20UpgradeSafe is Initializable, ContextUpgradeSafe, IERC20 {
34     using SafeMath for uint256;
35     using Address for address;
36
37     mapping (address => uint256) private _balances;
38
39     mapping (address => mapping (address => uint256)) private _allowances;
40
41     uint256 private _totalSupply;
42
43     string private _name;
44     string private _symbol;
45     uint8 private _decimals;
46

```

1 ERC20UpgradeSafe.sol

__gap variable from ERC20UpgradeSafe.sol is shadowing __gap variable of ContextUpgradeSafe.sol

<https://github.com/Base-Protocol/contracts-0.6.12/blob/cbbf4d8e1970e72df80b461358552c431296c6a2/contracts/ERC20UpgradeSafe.sol#L317>

```

297     function _setupDecimals(uint8 decimals_) internal {
298         _decimals = decimals_;
299     }
300
301     /**
302      * @dev Hook that is called before any transfer of tokens. This includes
303      * minting and burning.
304      *
305      * Calling conditions:
306      *
307      * - when `from` and `to` are both non-zero, `amount` of ``from``'s tokens
308      *   will be transferred to `to`.
309      * - when `from` is zero, `amount` tokens will be minted for `to`.
310      * - when `to` is zero, `amount` of ``from``'s tokens will be burned.
311      * - `from` and `to` are never both zero.
312      *
313      * To learn more about hooks, head to xref:ROOT:extending-contracts.adoc#using-hooks[Using Hooks].
314      */
315     function _beforeTokenTransfer(address from, address to, uint256 amount) internal virtual { }
316
317     uint256[44] private __gap;
318 }
319

```

3 ERC20UpgradeSafe.sol

3.5 STATIC ANALYSIS - LOW

Slither and MythX has been run on all the contracts
(BaseToken.sol, BaseTokenMonetaryPolicy.sol,
BaseTokenOrchestrator.sol, ERC20UpgradeSafe.sol, ERC677Token.sol
and Greeter.sol)

```

INFO:Detectors:
OwnableUpgradeSafe.__gap (FlattenBaseTokenOrchestrator.sol#185) shadows:
- ContextUpgradeSafe.__gap (FlattenBaseTokenOrchestrator.sol#104)
ERC20UpgradeSafe.__gap (FlattenBaseTokenOrchestrator.sol#2470) shadows:
- ContextUpgradeSafe.__gap (FlattenBaseTokenOrchestrator.sol#104)
BaseToken._totalSupply (FlattenBaseTokenOrchestrator.sol#2608) shadows:
- ERC20UpgradeSafe._totalSupply (FlattenBaseTokenOrchestrator.sol#2194)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
BaseTokenOrchestrator.addTransaction(address,bytes) (FlattenBaseTokenOrchestrator.sol#3215-3222) uses a Boolean constant improperly:
- transactionEnabled.push(true) (FlattenBaseTokenOrchestrator.sol#3219)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#misuse-of-a-boolean-constant
INFO:Detectors:
ERC20UpgradeSafe.__ERC20_init(string,string).name (FlattenBaseTokenOrchestrator.sol#2210) shadows:
- ERC20UpgradeSafe.name() (FlattenBaseTokenOrchestrator.sol#2228-2230) (function)
ERC20UpgradeSafe.__ERC20_init(string,string).symbol (FlattenBaseTokenOrchestrator.sol#2210) shadows:
- ERC20UpgradeSafe.symbol() (FlattenBaseTokenOrchestrator.sol#2236-2238) (function)
ERC20UpgradeSafe.__ERC20_init_unchained(string,string).name (FlattenBaseTokenOrchestrator.sol#2215) shadows:
- ERC20UpgradeSafe.name() (FlattenBaseTokenOrchestrator.sol#2228-2230) (function)
ERC20UpgradeSafe.__ERC20_init_unchained(string,string).symbol (FlattenBaseTokenOrchestrator.sol#2215) shadows:
- ERC20UpgradeSafe.symbol() (FlattenBaseTokenOrchestrator.sol#2236-2238) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Reentrancy in BaseTokenMonetaryPolicy.rebase() (FlattenBaseTokenOrchestrator.sol#2944-2984):
  External calls:
  - (mcap,mcapValid) = mcapOracle.getData() (FlattenBaseTokenOrchestrator.sol#2958)
  - (tokenPrice,tokenPriceValid) = tokenPriceOracle.getData() (FlattenBaseTokenOrchestrator.sol#2965)
  - supplyAfterRebase = BASE.rebase(epoch,supplyDelta) (FlattenBaseTokenOrchestrator.sol#2981)
  Event emitted after the call(s):
  - LogRebase(epoch,tokenPrice,mcap,supplyDelta,now) (FlattenBaseTokenOrchestrator.sol#2983)
Reentrancy in BaseTokenOrchestrator.rebase() (FlattenBaseTokenOrchestrator.sol#3191-3208):
  External calls:
  - policy.rebase() (FlattenBaseTokenOrchestrator.sol#3196)
  Event emitted after the call(s):
  - TransactionFailed(transactionDestination[i],i,transactionData[i]) (FlattenBaseTokenOrchestrator.sol#3203)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

```

```

INFO:Detectors:
BaseTokenMonetaryPolicy.rebase() (FlattenBaseTokenOrchestrator.sol#2944-2984) uses timestamp for comparisons
  Dangerous comparisons:
    - require(bool,string)(lastRebaseTimestampSec.add(minRebaseTimeIntervalSec) < now,cannot rebase yet) (FlattenBaseTokenOrchestrator.sol#2949)
BaseTokenMonetaryPolicy.inRebaseWindow() (FlattenBaseTokenOrchestrator.sol#3104-3109) uses timestamp for comparisons
  Dangerous comparisons:
    - (now.mod(minRebaseTimeIntervalSec) >= rebaseWindowOffsetSec && now.mod(minRebaseTimeIntervalSec) < (rebaseWindowOffsetSec.add(rebaseWindowLengthSec))) (FlattenBaseTokenOrchestrator.sol#3105-3108)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Initializable.isConstructor() (FlattenBaseTokenOrchestrator.sol#50-60) uses assembly
  - INLINE ASM (FlattenBaseTokenOrchestrator.sol#53)
console._sendLogPayload(bytes) (FlattenBaseTokenOrchestrator.sol#330-337) uses assembly
  - INLINE ASM (FlattenBaseTokenOrchestrator.sol#333-336)
Address.isContract(address) (FlattenBaseTokenOrchestrator.sol#2117-2126) uses assembly
  - INLINE ASM (FlattenBaseTokenOrchestrator.sol#2124)
ERC677Token.isContract(address) (FlattenBaseTokenOrchestrator.sol#2528-2537) uses assembly
  - INLINE ASM (FlattenBaseTokenOrchestrator.sol#2535)
BaseTokenOrchestrator.externalCall(address,bytes) (FlattenBaseTokenOrchestrator.sol#3274-3304) uses assembly
  - INLINE ASM (FlattenBaseTokenOrchestrator.sol#3279-3302)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
BaseToken.transfer(address,uint256) (FlattenBaseTokenOrchestrator.sol#2736-2749) compares to a boolean constant:
  - require(bool,string)(bannedUsers[msg.sender] == false,you are banned) (FlattenBaseTokenOrchestrator.sol#2742)
BaseToken.transferFrom(address,address,uint256) (FlattenBaseTokenOrchestrator.sol#2772-2788) compares to a boolean constant:
  - require(bool,string)(bannedUsers[msg.sender] == false,you are banned) (FlattenBaseTokenOrchestrator.sol#2778)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#boolean-equality

```

```

INFO:Detectors:
Different versions of Solidity is used in :
  - Version used: ['0.6.12', '>=0.4.22<0.8.0', '>=0.4.24<0.7.0', '^0.6.0', '^0.6.2']
  - >=0.4.24<0.7.0 (FlattenBaseTokenOrchestrator.sol#3)
  - ^0.6.0 (FlattenBaseTokenOrchestrator.sol#68)
  - ^0.6.0 (FlattenBaseTokenOrchestrator.sol#109)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#216)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#299)
  - >=0.4.22<0.8.0 (FlattenBaseTokenOrchestrator.sol#325)
  - ^0.6.0 (FlattenBaseTokenOrchestrator.sol#1863)
  - ^0.6.0 (FlattenBaseTokenOrchestrator.sol#1941)
  - ^0.6.2 (FlattenBaseTokenOrchestrator.sol#2094)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#2155)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#2475)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#2487)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#2496)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#2542)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#2852)
  - 0.6.12 (FlattenBaseTokenOrchestrator.sol#3151)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Pragma version<0.4.24<0.7.0 (FlattenBaseTokenOrchestrator.sol#3) allows old versions
Pragma version^0.6.0 (FlattenBaseTokenOrchestrator.sol#68) allows old versions
Pragma version^0.6.0 (FlattenBaseTokenOrchestrator.sol#109) allows old versions
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#216) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#299) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version<0.4.22<0.8.0 (FlattenBaseTokenOrchestrator.sol#325) is too complex
Pragma version^0.6.0 (FlattenBaseTokenOrchestrator.sol#1863) allows old versions
Pragma version^0.6.0 (FlattenBaseTokenOrchestrator.sol#1941) allows old versions
Pragma version^0.6.2 (FlattenBaseTokenOrchestrator.sol#2094) allows old versions
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#2155) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#2475) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#2487) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#2496) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#2542) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#2852) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
Pragma version0.6.12 (FlattenBaseTokenOrchestrator.sol#3151) necessitates a version too recent to be trusted. Consider deploying with 0.6.11
solc-0.6.12 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (FlattenBaseTokenOrchestrator.sol#2144-2150):
  - (success) = recipient.call{value: amount}() (FlattenBaseTokenOrchestrator.sol#2148)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

```

```

INFO:Detectors:
Variable Initializable.____gap (FlattenBaseTokenOrchestrator.sol#63) is not in mixedCase
Function ContextUpgradeSafe.__Context_init() (FlattenBaseTokenOrchestrator.sol#85-87) is not in mixedCase
Function ContextUpgradeSafe.__Context_init_unchained() (FlattenBaseTokenOrchestrator.sol#89-92) is not in mixedCase
Variable ContextUpgradeSafe.____gap (FlattenBaseTokenOrchestrator.sol#104) is not in mixedCase
Function OwnableUpgradeSafe.__Ownable_init() (FlattenBaseTokenOrchestrator.sol#133-136) is not in mixedCase
Function OwnableUpgradeSafe.__Ownable_init_unchained() (FlattenBaseTokenOrchestrator.sol#138-145) is not in mixedCase
Variable OwnableUpgradeSafe.____gap (FlattenBaseTokenOrchestrator.sol#185) is not in mixedCase
Contract console (FlattenBaseTokenOrchestrator.sol#327-1859) is not in CapWords
Function ERC20UpgradeSafe.__ERC20_init(string,string) (FlattenBaseTokenOrchestrator.sol#2210-2213) is not in mixedCase
Function ERC20UpgradeSafe.__ERC20_init_unchained(string,string) (FlattenBaseTokenOrchestrator.sol#2215-2222) is not in mixedCase
Variable ERC20UpgradeSafe.____gap (FlattenBaseTokenOrchestrator.sol#2470) is not in mixedCase
Parameter ERC677Token.transferAndCall(address,uint256,bytes)._to (FlattenBaseTokenOrchestrator.sol#2508) is not in mixedCase
Parameter ERC677Token.transferAndCall(address,uint256,bytes)._value (FlattenBaseTokenOrchestrator.sol#2508) is not in mixedCase
Parameter ERC677Token.transferAndCall(address,uint256,bytes)._data (FlattenBaseTokenOrchestrator.sol#2508) is not in mixedCase
Parameter ERC677Token.contractFallback(address,uint256,bytes)._to (FlattenBaseTokenOrchestrator.sol#2521) is not in mixedCase
Parameter ERC677Token.contractFallback(address,uint256,bytes)._value (FlattenBaseTokenOrchestrator.sol#2521) is not in mixedCase
Parameter ERC677Token.contractFallback(address,uint256,bytes)._data (FlattenBaseTokenOrchestrator.sol#2521) is not in mixedCase
Parameter ERC677Token.isContract(address)._addr (FlattenBaseTokenOrchestrator.sol#2528) is not in mixedCase
Parameter BaseTokenMonetaryPolicy.initialize(BaseToken,uint256).BASE_ (FlattenBaseTokenOrchestrator.sol#3080) is not in mixedCase
Variable BaseTokenMonetaryPolicy.BASE (FlattenBaseTokenOrchestrator.sol#2887) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformity-to-solidity-naming-conventions
INFO:Detectors:
console.slitherConstructorConstantVariables() (FlattenBaseTokenOrchestrator.sol#327-1859) uses literals with too many digits:
  - CONSOLE_ADDRESS = address(0x0000000000000000000000000000000000000000000000000000000000000000) (FlattenBaseTokenOrchestrator.sol#328)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

```



```

INFO:Detectors:
SafeMathInt.MAX_INT256 (FlattenBaseTokenOrchestrator.sol#225) is never used in SafeMathInt (FlattenBaseTokenOrchestrator.sol#223-295)
Initializeable.____gap (FlattenBaseTokenOrchestrator.sol#63) is never used in BaseToken (FlattenBaseTokenOrchestrator.sol#2560-2848)
OwnableUpgradeSafe.____gap (FlattenBaseTokenOrchestrator.sol#185) is never used in BaseToken (FlattenBaseTokenOrchestrator.sol#2560-2848)
Initializeable.____gap (FlattenBaseTokenOrchestrator.sol#63) is never used in BaseTokenMonetaryPolicy (FlattenBaseTokenOrchestrator.sol#2874-3147)
OwnableUpgradeSafe.____gap (FlattenBaseTokenOrchestrator.sol#185) is never used in BaseTokenMonetaryPolicy (FlattenBaseTokenOrchestrator.sol#2874-3147)
Initializeable.____gap (FlattenBaseTokenOrchestrator.sol#63) is never used in BaseTokenOrchestrator (FlattenBaseTokenOrchestrator.sol#3161-3305)
OwnableUpgradeSafe.____gap (FlattenBaseTokenOrchestrator.sol#185) is never used in BaseTokenOrchestrator (FlattenBaseTokenOrchestrator.sol#3161-3305)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
renounceOwnership() should be declared external:
- OwnableUpgradeSafe.renounceOwnership() (FlattenBaseTokenOrchestrator.sol#170-173)
transferOwnership(address) should be declared external:
- OwnableUpgradeSafe.transferOwnership(address) (FlattenBaseTokenOrchestrator.sol#179-183)
name() should be declared external:
- ERC20UpgradeSafe.name() (FlattenBaseTokenOrchestrator.sol#2228-2230)
symbol() should be declared external:
- ERC20UpgradeSafe.symbol() (FlattenBaseTokenOrchestrator.sol#2236-2238)
decimals() should be declared external:
- ERC20UpgradeSafe.decimals() (FlattenBaseTokenOrchestrator.sol#2253-2255)
totalSupply() should be declared external:
- BaseToken.totalSupply() (FlattenBaseTokenOrchestrator.sol#2708-2715)
- ERC20UpgradeSafe.totalSupply() (FlattenBaseTokenOrchestrator.sol#2260-2262)
balanceOf(address) should be declared external:
- BaseToken.balanceOf(address) (FlattenBaseTokenOrchestrator.sol#2721-2728)
- ERC20UpgradeSafe.balanceOf(address) (FlattenBaseTokenOrchestrator.sol#2267-2269)
allowance(address,address) should be declared external:
- BaseToken.allowance(address,address) (FlattenBaseTokenOrchestrator.sol#2757-2764)
- ERC20UpgradeSafe.allowance(address,address) (FlattenBaseTokenOrchestrator.sol#2287-2289)
approve(address,uint256) should be declared external:
- BaseToken.approve(address,uint256) (FlattenBaseTokenOrchestrator.sol#2801-2809)
- ERC20UpgradeSafe.approve(address,uint256) (FlattenBaseTokenOrchestrator.sol#2298-2301)
transferFrom(address,address,uint256) should be declared external:
- ERC20UpgradeSafe.transferFrom(address,address,uint256) (FlattenBaseTokenOrchestrator.sol#2315-2319)
- BaseToken.transferFrom(address,address,uint256) (FlattenBaseTokenOrchestrator.sol#2772-2788)
increaseAllowance(address,uint256) should be declared external:
- BaseToken.increaseAllowance(address,uint256) (FlattenBaseTokenOrchestrator.sol#2818-2826)
- ERC20UpgradeSafe.increaseAllowance(address,uint256) (FlattenBaseTokenOrchestrator.sol#2333-2336)
decreaseAllowance(address,uint256) should be declared external:
- BaseToken.decreaseAllowance(address,uint256) (FlattenBaseTokenOrchestrator.sol#2834-2847)
- ERC20UpgradeSafe.decreaseAllowance(address,uint256) (FlattenBaseTokenOrchestrator.sol#2352-2355)
transferAndCall(address,uint256,bytes) should be declared external:
- ERC677Token.transferAndCall(address,uint256,bytes) (FlattenBaseTokenOrchestrator.sol#2508-2519)
- ERC677.transferAndCall(address,uint256,bytes) (FlattenBaseTokenOrchestrator.sol#2480)
onTokenTransfer(address,uint256,bytes) should be declared external:
- ERC677Receiver.onTokenTransfer(address,uint256,bytes) (FlattenBaseTokenOrchestrator.sol#2491)
initialize() should be declared external:
- BaseToken.initialize() (FlattenBaseTokenOrchestrator.sol#2672-2691)
setUserBanStatus(address,bool) should be declared external:
- BaseToken.setUserBanStatus(address,bool) (FlattenBaseTokenOrchestrator.sol#2693-2703)
initialize(BaseToken,uint256) should be declared external:
- BaseTokenMonetaryPolicy.initialize(BaseToken,uint256) (FlattenBaseTokenOrchestrator.sol#3080-3098)
initialize(address) should be declared external:
- BaseTokenOrchestrator.initialize(address) (FlattenBaseTokenOrchestrator.sol#3175-3181)
Reference: https://github.com/cryptic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external

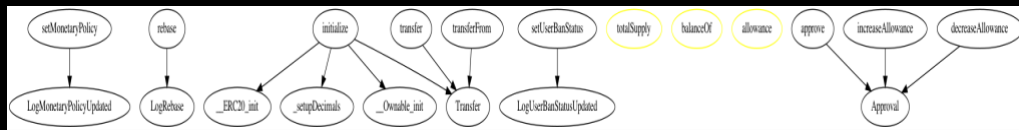
```

MythX detected 0 **High** findings, 3 **Medium**, and 7 **Low**.

0 High		3 Medium		7 Low	
ID	SEVERITY	NAME	FILE	LOCATION	
SWC-128	Medium	Loop over unbounded data structure.	BaseTokenOrchestrator.sol	L: 50 C: 25	
SWC-128	Medium	Implicit loop over unbounded data structure.	BaseTokenOrchestrator.sol	L: 63 C: 70	
SWC-128	Medium	Implicit loop over unbounded data structure.	BaseTokenOrchestrator.sol	L: 89 C: 12	
SWC-108	Low	State variable visibility is not set.	BaseTokenOrchestrator.sol	L: 18 C: 11	
SWC-108	Low	State variable visibility is not set.	BaseTokenOrchestrator.sol	L: 19 C: 14	
SWC-108	Low	State variable visibility is not set.	BaseTokenOrchestrator.sol	L: 20 C: 12	
SWC-115	Low	Use of 'tx.origin' as a part of authorization control.	BaseTokenOrchestrator.sol	L: 46 C: 30	
SWC-131	Low	Unused function parameter 'from'.	ERC20UpgradeSafe.sol	L: 315 C: 34	
SWC-131	Low	Unused function parameter 'to'.	ERC20UpgradeSafe.sol	L: 315 C: 48	
SWC-131	Low	Unused function parameter 'amount'.	ERC20UpgradeSafe.sol	L: 315 C: 60	

3.6 SOLGRAPH - INFORMATIONAL

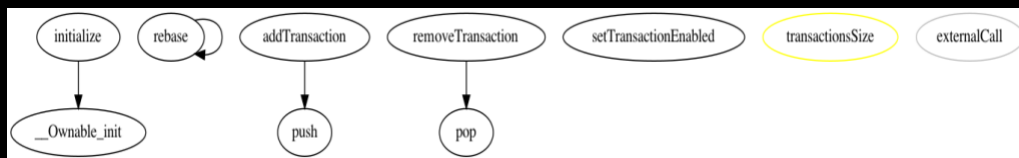
BaseToken.sol



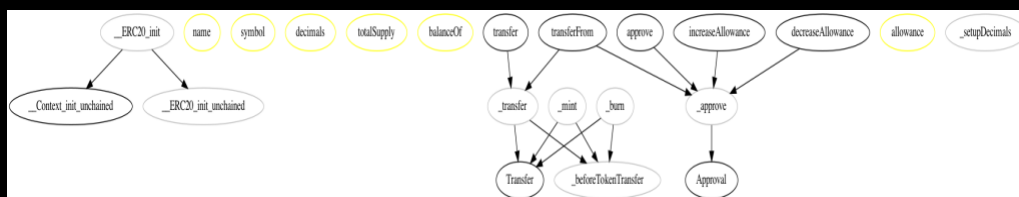
BaseTokenMonetaryPolicy.sol



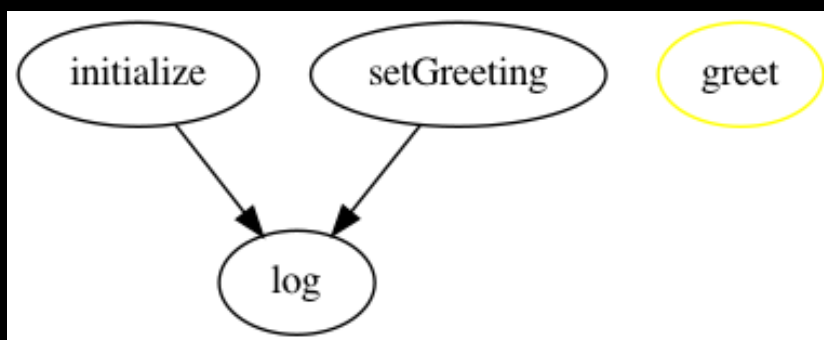
BaseTokenOrchestrator.sol



ERC20UpgradeSafe.sol



Greeter.sol





THANK YOU FOR CHOOSING

 **HALBORN**