SolidityScan | Powered by CRED SHIELDS

Security Assessment Report

BTA Bitarena

12 Feb 2025

This security assessment report was prepared by SolidityScan.com, a cloud-based Smart Contract Scanner.

Table of Contents

01 Vulnerability Classification and Severity

02 Executive Summary

03 Findings Summary

04 Vulnerability Details

- USE OF TX.GASPRICE
- BLOCK.NUMBER INCONSISTENCIES ON VARIOUS L2S
- STRICT EQUALITY CHECK IN BLOCK.TIMESTAMP
- MISSING EVENTS
- OUTDATED COMPILER VERSION
- USE OWNABLE2STEP
- BLOCK VALUES AS A PROXY FOR TIME
- IF-STATEMENT REFACTORING
- MISSING INDEXED KEYWORDS IN EVENTS
- MISSING PAYABLE IN CALL FUNCTION
- MISSING UNDERSCORE IN NAMING VARIABLES
- UNUSED RECEIVE FALLBACK
- USE SCIENTIFIC NOTATION
- AVOID RE-STORING VALUES
- CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE
- CHEAPER INEQUALITIES IN IF()
- CODE OPTIMIZATION BY USING MAX AND MIN

CONSTANT STATE VARIABLES
DEFINE CONSTRUCTOR AS PAYABLE
FUNCTIONS CAN BE IN-LINED
REVERTING FUNCTIONS CAN BE PAYABLE
LONG NUMBER LITERALS
MULTIPLICATION/DIVISION BY TWO SHOULD USE BIT SHIFTING
NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT
OPTIMIZING ADDRESS ID MAPPING
PUBLIC CONSTANTS CAN BE PRIVATE
STORAGE VARIABLE CACHING IN MEMORY
SUPERFLUOUS EVENT FIELDS
USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

05 Scan History

06 Disclaimer

01. Vulnerability Classification and Severity

Description

To enhance navigability, the document is organized in descending order of severity for easy reference. Issues are categorized as *Fixed*, *Pending Fix*, or *Won't Fix*, indicating their current status. *Won't Fix* denotes that the team is aware of the issue but has chosen not to resolve it. Issues labeled as *Pending Fix* state that the bug is yet to be resolved. Additionally, each issue's severity is assessed based on the risk of exploitation or the potential for other unexpected or unsafe behavior.

Critical

The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.

High

High-severity vulnerabilities pose a significant risk to both the Smart Contract and the organization. They can lead to user fund losses, may have conditional requirements, and are challenging to exploit.

Medium

The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.

Low

The issue has minimal impact on the contract's ability to operate.

Informational

The issue does not affect the contract's operational capability but is considered good practice to address.

Gas

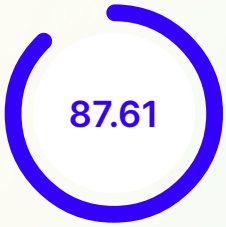
This category deals with optimizing code and refactoring to conserve gas.

02. Executive Summary

BTA Bitarena

Uploaded Solidity File(s)

Language	Audit Methodology	Website
Solidity	**Static Scanning**	-
Publishers/Owner Name	Organization	Contact Email
-	-	-



Security Score is GREAT

The SolidityScan score is calculated based on lines of code and weights assigned to each issue depending on the severity and confidence. To improve your score, view the detailed result and leverage the remediation solutions provided.

This report has been prepared for BTA Bitarena using SolidityScan to scan and discover vulnerabilities and safe coding practices in their smart contract including the libraries used by the contract that are not officially recognized. The SolidityScan tool runs a comprehensive static analysis on the Solidity code and finds vulnerabilities ranging from minor gas optimizations to major vulnerabilities leading to the loss of funds. The coverage scope pays attention to all the informational and critical vulnerabilities with over (100+) modules. The scanning and auditing process covers the following areas:

Various common and uncommon attack vectors will be investigated to ensure that the smart contracts are secure from malicious actors. The scanner modules find and flag issues related to Gas optimizations that help in reducing the overall Gas cost It scans and evaluates the codebase against industry best practices and standards to ensure compliance It makes sure that the officially recognized libraries used in the code are secure and up to date.

The SolidityScan Team recommends running regular audit scans to identify any vulnerabilities that are introduced after BTA Bitarena introduces new features or refactors the code.

03. Findings Summary

BTA Bitarena
File Scan

Security Score	Scan duration	Lines of code
87.61/100	**4 secs**	**823**

0	1	3	11	23	44
Crit	High	Med	Low	Info	Gas

This audit report has not been verified by the SolidityScan team. To learn more about our published reports. click here

ACTION TAKEN

0	0	0	82
Fixed	*False Positive*	*Won't Fix*	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
H001	High	USE OF TX.GASPRICE	1	Automated	*Pending Fix*
M001	Medium	BLOCK.NUMBER INCONSISTENCIES ON VARIOUS L2S	2	Automated	*Pending Fix*
M002	Medium	STRICT EQUALITY CHECK IN BLOCK.TIMESTAMP	1	Automated	*Pending Fix*
L001	Low	MISSING EVENTS	9	Automated	*Pending Fix*
L002	Low	OUTDATED COMPILER VERSION	1	Automated	*Pending Fix*
L003	Low	USE OWNABLE2STEP	1	Automated	*Pending Fix*
I001	Informational	BLOCK VALUES AS A PROXY FOR TIME	1	Automated	*Pending Fix*
I002	Informational	IF-STATEMENT REFACTORING	1	Automated	*Pending Fix*
I003	Informational	MISSING INDEXED KEYWORDS IN EVENTS	5	Automated	*Pending Fix*
I004	Informational	MISSING PAYABLE IN CALL FUNCTION	1	Automated	*Pending Fix*
I005	Informational	MISSING UNDERSCORE IN NAMING VARIABLES	10	Automated	*Pending Fix*
I006	Informational	UNUSED RECEIVE FALLBACK	2	Automated	*Pending Fix*
I007	Informational	USE SCIENTIFIC NOTATION	3	Automated	*Pending Fix*
G001	Gas	AVOID RE-STORING VALUES	7	Automated	*Pending Fix*

S. No.	Severity	Bug Type	Instances	Detection Method	Status
G002	Gas	CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE	5	Automated	*Pending Fix*
G003	Gas	CHEAPER INEQUALITIES IN IF()	2	Automated	*Pending Fix*
G004	Gas	CODE OPTIMIZATION BY USING MAX AND MIN	1	Automated	*Pending Fix*
G005	Gas	CONSTANT STATE VARIABLES	2	Automated	*Pending Fix*
G006	Gas	DEFINE CONSTRUCTOR AS PAYABLE	1	Automated	*Pending Fix*
G007	Gas	FUNCTIONS CAN BE IN-LINED	1	Automated	*Pending Fix*
G008	Gas	REVERTING FUNCTIONS CAN BE PAYABLE	8	Automated	*Pending Fix*
G009	Gas	LONG NUMBER LITERALS	1	Automated	*Pending Fix*
G010	Gas	MULTIPLICATION/DIVISION BY TWO SHOULD USE BIT SHIFTING	1	Automated	*Pending Fix*
G011	Gas	NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT	1	Automated	*Pending Fix*
G012	Gas	OPTIMIZING ADDRESS ID MAPPING	4	Automated	*Pending Fix*
G013	Gas	PUBLIC CONSTANTS CAN BE PRIVATE	1	Automated	*Pending Fix*
G014	Gas	STORAGE VARIABLE CACHING IN MEMORY	12	Automated	*Pending Fix*
G015	Gas	SUPERFLUOUS EVENT FIELDS	1	Automated	*Pending Fix*
G016	Gas	USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE	2	Automated	*Pending Fix*

04. Vulnerability Details

Issue Type

USE OF TX.GASPRICE

S. No.	Severity	Detection Method	Instances
H001	High	Automated	1

Upgrade your Plan to view the full report

1 High Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

BLOCK.NUMBER INCONSISTENCIES ON VARIOUS L2S

S. No.	Severity	Detection Method	Instances
M001	Medium	Automated	2

Bug ID	File Location	Line No.	Action Taken

Upgrade your Plan to view the full report

2 Medium Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

MISSING EVENTS

S. No.	Severity	Detection Method	Instances
L001	Low	Automated	9

Bug ID	File Location	Line No.	Action Taken

Upgrade your Plan to view the full report

9 Low Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

BLOCK VALUES AS A PROXY FOR TIME

S. No.	Severity	Detection Method	Instances
I001	Informational	Automated	1

Bug ID	File Location	Line No.	Action Taken

Upgrade your Plan to view the full report

1 Informational Issues Found

Please upgrade your plan to view all the issues in your report.

Upgrade

Issue Type

AVOID RE-STORING VALUES

S. No.	Severity	Detection Method	Instances
G001	Gas	Automated	7

Description

The function is found to be allowing re-storing the value in the contract's state variable even when the old value is equal to the new value. This practice results in unnecessary gas consumption due to the `Gsreset` operation (2900 gas), which could be avoided. If the old value and the new value are the same, not updating the storage would avoid this cost and could instead incur a `Gcoldsload` (2100 gas) or a `Gwarmaccess` (100 gas), potentially saving gas.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_7	/BitarenaToken (2).sol	L492 - L599	*Pending Fix*

/BitarenaToken (2).sol L492 - L599

```
491     ///   □ 'true' IF swap type is detected
492     function swapDetectType
493     (
494       address sender,
495       address recipient,
496       string memory swapType
497     )
498     internal
499     returns
500     (
501       bool
502     )
503     {
504
505       // ┌──────────────
```

/BitarenaToken (2).sol L492 - L599

```
    // ├──────────────────────────────────────────────────────────────────────────┤
    // │ 1. 'UniversalRouter' = is 'msg.sender'                                   │
    // │ 2. 'UniversalRouter' = is 'sender/from'                                  │
    // │ 3. 'user'            = is 'recipient/to'                                 │
    // ├──────────────────────────────────────────────────────────────────────────┤
    // │ CONDTION [1]                                                             │
    // │  Description                                                             │
    // │  :: Post-Swap Detection                                                  │
    // │  :: Typically combined with CONDTION [0], but can be a standalone        │
    // │  :: **WORKING** condition                                                │
    // ├──────────────────────────────────────────────────────────────────────────┤
    // │ 1. 'PancakeV3Pool|UniswapV3Pool' = is 'msg.sender'                       │
    // │ 2. 'PancakeV3Pool|UniswapV3Pool' = is 'sender/from'                      │
    // │ 3. 'user'                        = is 'recipient/to'                     │
    // │ 4. 'user'                        = is 'tx.origin'                        │
    // └──────────────────────────────────────────────────────────────────────────┘

    if
    (
      swapType.equal("buy")
      && instanceFeeLogic.numBuyFee != 0
      && instanceFeeLogic.buyCondition == 0
    )
    {
      string memory mapBuyKey = string.concat(Strings.toString(block.number), "_", Strings.toString(block.timestamp), "_", Strings.toString(tx.gasprice));

      // [□]
      if (isDebugActive) emit DebugTransactionSwapCheckpoint(mapBuyKey);

      // ┌──────
      // │ NOTE:
      // │ |: Validating the (pre-swap) 'Buy' condition
      // └──────
      if
      (
        msg.sender == sender
        && isUniswapV3Pool(sender)
        && mapSwapContext[tx.origin][mapBuyKey] == address(0)
      )
      {
        // [□]
        if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-0]");

        mapSwapContext[tx.origin][mapBuyKey] = sender;
```

/BitarenaToken (2).sol L492 - L599

```
553
554         mapSwapContext[tx.origin][mapBuyKey] = sender;
555         return false;
556       }
557       // ╭──────
558       // │ NOTE:
559       // │ |: Validating the (post-swap) 'Buy' condition
560       // ╰──────
561       else if
562       (
563         tx.origin == recipient
564         && msg.sender == instanceFeeLogic.adrUniswapUniversalRouter
565         && sender == instanceFeeLogic.adrUniswapUniversalRouter
566         && mapSwapContext[tx.origin][mapBuyKey] != address(0)
567         && !mapAdrExcludedForBuy[(mapSwapContext[tx.origin][mapBuyKey])]
568       )
569       {
570         // [□]
571         if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-1]");
572
573         delete mapSwapContext[tx.origin][mapBuyKey];
574         return true;
575       }
576
577       return false;
578     }
579
580     // ╭──────────────────────────────────────────────────────────────────────────────────╮
581     // │ □ | SELL                                                                         |
582     // ├──────────────────────────────────────────────────────────────────────────────────┤
583     // │ CONDTION [1]                                                                     |
584     // │ 1. 'Permit2'                    = is 'msg.sender'                                |
585     // │ 2. 'user'                       = is 'sender/from'                               |
586     // │ 3. 'PancakeV3Pool|UniswapV3Pool' = is 'recipient/to'                             |
587     // ╰──────────────────────────────────────────────────────────────────────────────────╯
588
589     if
590     (
591       swapType.equal("sell")
592       && instanceFeeLogic.numSellFee != 0
593       && msg.sender == instanceFeeLogic.adrUniswapPermit2
594       && !isUniswapV3Pool(sender)
595       && !mapAdrExcludedForSell[recipient]
596     ) return true;
```

/BitarenaToken (2).sol L492 - L599

```
595         && !mapAdrExcludedForSell[recipient]
596       ) return true;
597
598       return false;
599     }
600
601     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_8	/BitarenaToken (2).sol	L801 - L817	⚠️ *Pending Fix*

/BitarenaToken (2).sol L801 - L817

```
800     ///   type of exclusion (buy/sell/general)
801     function setToggleExcludeAddressFromFee
802     (
803       address account,
804       bool isExcluded,
805       string memory exclusionType
806     )
807     external
808     onlyOwner
809     {
810       if (exclusionType.equal("sell"))
811         mapAdrExcludedForSell[account] = isExcluded;
812       else if (exclusionType.equal("buy"))
813         mapAdrExcludedForBuy[account] = isExcluded;
814       else if (exclusionType.equal("general"))
815         mapAddressExcluded[account] = isExcluded;
816       return;
817     }
818
819     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_9	/BitarenaToken (2).sol	L825 - L836	⚠ *Pending Fix*

/BitarenaToken (2).sol L825 - L836

```
824     ///   address of UiswapUniversalRouter
825     function setUniswapOfficialAddress
826     (
827       address _adrUniswapPermit2,
828       address _adrUniswapUniversalRouter
829     )
830     external
831     onlyOwner
832     {
833       instanceFeeLogic.adrUniswapPermit2 = _adrUniswapPermit2;
834       instanceFeeLogic.adrUniswapUniversalRouter = _adrUniswapUniversalRouter;
835       return;
836     }
837
838     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_10	/BitarenaToken (2).sol	L842 - L851	*Pending Fix*

/BitarenaToken (2).sol L842 - L851

```
///   condition to set
function setBuyCondition
(
  uint256 condition
)
external
onlyOwner
{
  instanceFeeLogic.buyCondition = condition;
  return;
}

/// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_11	/BitarenaToken (2).sol	L872 - L882	*Pending Fix*

/BitarenaToken (2).sol L872 - L882

```
///   address of the Uniswap V3 Pool Created
function setMainLiquidtyPool
(
  address account
)
external
onlyOwner
{
  mapAddressV3Pool[account] = true;
  instanceFeeLogic.adrBtaUsdPool = account;
  return;
}

/// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_12	/BitarenaToken (2).sol	L888 - L903	⚠ *Pending Fix*

/BitarenaToken (2).sol L888 - L903

```
887     ///   buy fee expressed as USD (fiat)
888     function setSwapFees
889     (
890       uint256 fee,
891       string memory feeType
892     )
893     external
894     onlyOwner
895     {
896       if (fee < 0 || fee > 250)
897         revert ErrorGeneric(fee, "BTA :: fee cannot be lower than 0 or higher than 2.5 USD
      (=250)");
898       if (feeType.equal("buy"))
899         instanceFeeLogic.numBuyFee = fee;
900       else if (feeType.equal("sell"))
901         instanceFeeLogic.numSellFee = fee;
902       return;
903     }
904
905     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_13	/BitarenaToken (2).sol	L909 - L918	⚠ *Pending Fix*

/BitarenaToken (2).sol L909 - L918

```
908     ///   □ flag to set
909     function setDebugging
910     (
911       bool _isDebugActive
912     )
913     external
914     onlyOwner
915     {
916       isDebugActive = _isDebugActive;
917       return;
918     }
919
920     // #endregion ➤ □ GETTER/SETTER
```

Issue Type

CACHE ADDRESS(THIS) WHEN USED MORE THAN ONCE

S. No.	Severity	Detection Method	Instances
G002	Gas	Automated	5

Description

The repeated usage of `address(this)` within the contract could result in increased gas costs due to multiple exec utions of the same computation, potentially impacting efficiency and overall transaction expenses.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_1	/BitarenaToken (2).sol	L240 - L240	*Pending Fix*

/BitarenaToken (2).sol L240 - L240

```
239     mapAddressExcluded[owner()]                                  = true;
240     mapAddressExcluded[address(this)]                            = true;
241     mapAddressExcluded[instanceBetarenaAddresses.adrFoundingTeam]  = true;
242     mapAddressExcluded[instanceBetarenaAddresses.adrAdvisoryBoard] = true;
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_2	/BitarenaToken (2).sol	L285 - L285	*Pending Fix*

/BitarenaToken (2).sol L285 - L285

```
284     {
285       return address(this).balance;
286     }
287
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_3	/BitarenaToken (2).sol	L296 - L296	*Pending Fix*

/BitarenaToken (2).sol L296 - L296

```
295
296       uint256 amount = address(this).balance;
297       address payable recipient = payable(owner());
298       (bool success,) = recipient.call{ value: amount }("BTA :: Amount Withdrawn from smart
    contract");
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_4	/BitarenaToken (2).sol	L746 - L746	*Pending Fix*

/BitarenaToken (2).sol L746 - L746

```
745        // └────
746        if (token0 == address(this))
747        {
748          sqrtPriceX96Pow = uint256(sqrtPriceX96 * 10**12);
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_5	/BitarenaToken (2).sol	L750 - L750	*Pending Fix*

/BitarenaToken (2).sol L750 - L750

```
749        }
750        else if (token1 == address(this))
751        {
752          sqrtPriceX96Pow = uint256(sqrtPriceX96);
```

Issue Type

CHEAPER INEQUALITIES IN IF()

S. No.	Severity	Detection Method	Instances
G003	Gas	Automated	2

Description

The contract was found to be doing comparisons using inequalities inside the if statement.
When inside the `if` statements, non-strict inequalities `(>=, <=)` are usually cheaper than the strict equalities `(>, <)`.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_69	/BitarenaToken (2).sol	L656 - L656	*Pending Fix*

/BitarenaToken (2).sol L656 - L656

```
655
656     if (numTotalSupply < numNonCirculatingSupply)
657     {
658       numCirculatingSupply = 0;
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_70	/BitarenaToken (2).sol	L896 - L896	⚠ *Pending Fix*

/BitarenaToken (2).sol L896 - L896

```
895     {
896       if (fee < 0 || fee > 250)
897         revert ErrorGeneric(fee, "BTA :: fee cannot be lower than 0 or higher than 2.5 USD
      (=250)");
898       if (feeType.equal("buy"))
```

Issue Type

CODE OPTIMIZATION BY USING MAX AND MIN

S. No.	Severity	Detection Method	Instances
G004	Gas	Automated	1

Description

The contract was using the expression to calculate the maximum storage capacity of the data type. This both lacks r eadability and costs more gas.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_34	/BitarenaToken (2).sol	L759 - L759	*Pending Fix*

/BitarenaToken (2).sol L759 - L759

```
758
759         uint256 price = sqrtPriceX96Pow / 2**96;
760         price = price**2;
761         price = price * 10**6;
```

Issue Type

CONSTANT STATE VARIABLES

S. No.	Severity	Detection Method	Instances
G005	Gas	Automated	2

Description

The contract has defined state variables whose values are never modified throughout the contract.
The variables whose values never change should be marked as `constant` to save gas.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_32	/BitarenaToken (2).sol	L129 - L129	*Pending Fix*

/BitarenaToken (2).sol L129 - L129

```
128
129     IBetarenaAddresses public instanceBetarenaAddresses;
130     IFeeLogic public instanceFeeLogic;
131
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_33	/BitarenaToken (2).sol	L130 - L130	*Pending Fix*

/BitarenaToken (2).sol L130 - L130

```
129     IBetarenaAddresses public instanceBetarenaAddresses;
130     IFeeLogic public instanceFeeLogic;
131
132     // #endregion ➤ □ VARIABLES
```

Issue Type

DEFINE CONSTRUCTOR AS PAYABLE

S. No.	Severity	Detection Method	Instances
G006	Gas	Automated	1

Description

Developers can save around 10 opcodes and some gas if the constructors are defined as payable.
However, it should be noted that it comes with risks because payable constructors can accept ETH during deployment.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_59	/BitarenaToken (2).sol	L170 - L255	*Pending Fix*

/BitarenaToken (2).sol L170 - L255

```
169     ///   BitarenaToken
170     constructor
171     (
172       string memory _name,
173       string memory _symbol,
174
175       address _adrFoundingTeam,
176       address _adrAdvisoryBoard,
177       address _adrInvestors,
178       address _adrTeam,
179       address _adrParticipants,
180       address _adrMarketing,
181       address _adrLiquidity,
182       address _adrReserve,
183
184       address _adrFeeAddress,
185       address _adrUniswapPermit2,
186       address _adrUniswapUniversalRouter
187     )
188     ERC20(_name, _symbol)
189     ERC20Permit(_name)
190     Ownable(msg.sender)
```

/BitarenaToken (2).sol L170 - L255

```
    ERC20Permit(_name)
    Ownable(msg.sender)
    {
      instanceBetarenaAddresses.adrFoundingTeam  = _adrFoundingTeam;
      instanceBetarenaAddresses.adrAdvisoryBoard = _adrAdvisoryBoard;
      instanceBetarenaAddresses.adrInvestors     = _adrInvestors;
      instanceBetarenaAddresses.adrTeam          = _adrTeam;
      instanceBetarenaAddresses.adrParticipants  = _adrParticipants;
      instanceBetarenaAddresses.adrMarketing     = _adrMarketing;
      instanceBetarenaAddresses.adrLiquidity     = _adrLiquidity;
      instanceBetarenaAddresses.adrReserve       = _adrReserve;

      instanceFeeLogic.adrFeeDeposit             = _adrFeeAddress;
      instanceFeeLogic.adrUniswapPermit2         = _adrUniswapPermit2;
      instanceFeeLogic.adrUniswapUniversalRouter = _adrUniswapUniversalRouter;

      // [□]
      // solhint-disable no-console
      // console.log(unicode"□ [checkpoint] :: initialize(..)");
      // console.log(unicode"□ [var] _adrFoundingTeam :: %s", _adrFoundingTeam);
      // console.log(unicode"□ [var] _adrAdvisoryBoard :: %s", _adrAdvisoryBoard);
      // console.log(unicode"□ [var] _adrInvestors :: %s", _adrInvestors);
      // console.log(unicode"□ [var] _adrTeam :: %s", _adrTeam);
      // console.log(unicode"□ [var] _adrParticipants :: %s", _adrParticipants);
      // console.log(unicode"□ [var] _adrMarketing :: %s", _adrMarketing);
      // console.log(unicode"□ [var] _adrLiquidity :: %s", _adrLiquidity);
      // console.log(unicode"□ [var] _adrReserve :: %s", _adrReserve);
      // console.log(unicode"□ [var] _adrUniswapPermit2 :: %s", _adrUniswapPermit2);
      // console.log(unicode"□ [var] address(this) :: %s", address(this));
      // solhint-enable no-console

      // ┌──────────────────────────────────────────────────────────────────────────────┐
      // │ □ │ MAIN INTIALIZER LOGIC                                                    │
      // └──────────────────────────────────────────────────────────────────────────────┘

      uint256 _onePercent = calcOnePercentOfTotalSupply();

      _mint(instanceBetarenaAddresses.adrFoundingTeam,  (_onePercent * 5));
      _mint(instanceBetarenaAddresses.adrAdvisoryBoard, (_onePercent * 2));
      _mint(instanceBetarenaAddresses.adrInvestors,     (_onePercent * 4));
      _mint(instanceBetarenaAddresses.adrTeam,          (_onePercent * 5));
      _mint(instanceBetarenaAddresses.adrParticipants,  (_onePercent * 10));
      _mint(instanceBetarenaAddresses.adrMarketing,     (_onePercent * 10));
      _mint(instanceBetarenaAddresses.adrLiquidity,     (_onePercent * 60));
```

/BitarenaToken (2).sol L170 - L255

```
231     _mint(instanceBetarenaAddresses.adrMarketing,     (_onePercent * 10));
232     _mint(instanceBetarenaAddresses.adrLiquidity,     (_onePercent * 60));
233     _mint(instanceBetarenaAddresses.adrReserve,       (_onePercent * 4));
234
235     // Causes Error:
236     // ProviderError: Error: VM Exception while processing transaction: reverted with panic code 0
    x11 (Arithmetic operation overflowed outside of an unchecked block)
237     updateCirculatingSupply();
238
239     mapAddressExcluded[owner()]                                  = true;
240     mapAddressExcluded[address(this)]                            = true;
241     mapAddressExcluded[instanceBetarenaAddresses.adrFoundingTeam]  = true;
242     mapAddressExcluded[instanceBetarenaAddresses.adrAdvisoryBoard] = true;
243     mapAddressExcluded[instanceBetarenaAddresses.adrInvestors]     = true;
244     mapAddressExcluded[instanceBetarenaAddresses.adrTeam]          = true;
245     mapAddressExcluded[instanceBetarenaAddresses.adrParticipants]  = true;
246     mapAddressExcluded[instanceBetarenaAddresses.adrMarketing]     = true;
247     mapAddressExcluded[instanceBetarenaAddresses.adrLiquidity]     = true;
248     mapAddressExcluded[instanceBetarenaAddresses.adrReserve]       = true;
249
250     instanceFeeLogic.numBuyFee  = 100;
251     instanceFeeLogic.numSellFee = 50;
252     instanceFeeLogic.buyCondition = 0;
253
254     return;
255   }
256
257   // #region ➤ □ METHODS
```

Issue Type

FUNCTIONS CAN BE IN-LINED

S. No.	Severity	Detection Method	Instances
G007	Gas	Automated	1

Description

The internal function was called only once throughout the contract. Internal functions cost more gas due to additional `JUMP` instructions and stack operations.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_27	/BitarenaToken (2).sol	L391 - L480	*Pending Fix*

/BitarenaToken (2).sol L391 - L480

```
390     ///   amount to transfer
391     function transferBuySellTakeFees
392     (
393       address sender,
394       address recipient,
395       uint256 amount
396     )
397     internal
398     {
399       // ┌──────
400       // │ CHECK |: Preliminary check (w/ exit)
401       // ├──────
402       // │ 1. Check if address is excluded from general fee
403       // │ 2. Check if address is 0x0 (a void)
404       // └──────
405       if
406       (
407         (
408           mapAddressExcluded[sender]
409           || mapAddressExcluded[recipient]
410         )
411         ||
```

/BitarenaToken (2).sol L391 - L480

```
        )
        ||
        (
          sender == address(0)
          || recipient == address(0)
        )
      )
      {
        // [⌐]
        if (isDebugActive) emit DebugTransactionStandard(sender, recipient);
        return;
      }

      // ┌──────
      // │ CHECK |: Buy-Action & Fee Gathering
      // └──────
      if (swapDetectType(sender, recipient, "buy"))
      {
        // [⌐]
        if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numBuyFee, "buy");

        uint256 buyFeeAmount;
        uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

        unchecked
        {
          buyFeeAmount = calculateBitarenaFee(instanceFeeLogic.numBuyFee, priceBtaFor1Usd);
          if (buyFeeAmount >= amount)
          {
            buyFeeAmount = 0;
            return;
          }
        }

        // ┌──────
        // │ NOTE:
        // │ ➤ 1XY% of the buyFeeAmount is taken
        // └──────
        super._update(recipient, instanceFeeLogic.adrFeeDeposit, buyFeeAmount);
      }

      // ┌──────
      // │ CHECK |: Sell-Action & Fee Gathering
      // └──────
```

/BitarenaToken (2).sol L391 - L480

```
    // │ CHECK |: Sell-Action & Fee Gathering
    // └──────
    if (swapDetectType(sender, recipient, "sell"))
    {
      // [⌷]
      if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numSellFee, "sell");

      uint256 sellFeeAmount;
      uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

      unchecked
      {
        sellFeeAmount = calculateBitarenaFee(instanceFeeLogic.numSellFee, priceBtaFor1Usd);
        if (amount + sellFeeAmount >= balanceOf(sender))
        {
          sellFeeAmount = 0;
          return;
        }
      }

      // ┌──────
      // │ NOTE:
      // │ ➤ 1XY% of the sellFeeAmount is taken
      // └──────
      super._update(sender, instanceFeeLogic.adrFeeDeposit, sellFeeAmount);
    }

    return;
  }

  /// @notice
```

Issue Type

REVERTING FUNCTIONS CAN BE PAYABLE

S. No.	Severity	Detection Method	Instances
G008	Gas	Automated	8

Description

If a function modifier such as `onlyOwner` is used, the function will revert if a normal user tries to pay the function. Marking the function as payable will lower the gas cost for legitimate callers because the compiler will not include checks for whether a payment was provided.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_19	/BitarenaToken (2).sol	L290 - L303	*Pending Fix*

/BitarenaToken (2).sol L290 - L303

```
289     ///   □ withdraws the locked amount of ETH
290     function withdrawETH()
291     external
292     nonReentrant
293     onlyOwner
294     {
295
296       uint256 amount = address(this).balance;
297       address payable recipient = payable(owner());
298       (bool success,) = recipient.call{ value: amount }("BTA :: Amount Withdrawn from smart
contract");
299       if (!success) revert ErrorGeneric (0, "BTA :: Withdraw transfer failed");
300       // [□]
301       if (isDebugActive) emit DebugFunctionWithdrawETH(owner(), amount);
302       return;
303     }
304
305     //
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_20	/BitarenaToken (2).sol	L801 - L817	*Pending Fix*

/BitarenaToken (2).sol

L801 - L817

```
800     ///   type of exclusion (buy/sell/general)
801     function setToggleExcludeAddressFromFee
802     (
803       address account,
804       bool isExcluded,
805       string memory exclusionType
806     )
807     external
808     onlyOwner
809     {
810       if (exclusionType.equal("sell"))
811         mapAdrExcludedForSell[account] = isExcluded;
812       else if (exclusionType.equal("buy"))
813         mapAdrExcludedForBuy[account] = isExcluded;
814       else if (exclusionType.equal("general"))
815         mapAddressExcluded[account] = isExcluded;
816       return;
817     }
818
819     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_21	/BitarenaToken (2).sol	L825 - L836	⚠ *Pending Fix*

/BitarenaToken (2).sol L825 - L836

```
824     ///   address of UiswapUniversalRouter
825     function setUniswapOfficialAddress
826     (
827       address _adrUniswapPermit2,
828       address _adrUniswapUniversalRouter
829     )
830     external
831     onlyOwner
832     {
833       instanceFeeLogic.adrUniswapPermit2 = _adrUniswapPermit2;
834       instanceFeeLogic.adrUniswapUniversalRouter = _adrUniswapUniversalRouter;
835       return;
836     }
837
838     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_22	/BitarenaToken (2).sol	L842 - L851	*Pending Fix*

/BitarenaToken (2).sol L842 - L851

```
841     ///   □ condition to set
842     function setBuyCondition
843     (
844       uint256 condition
845     )
846     external
847     onlyOwner
848     {
849       instanceFeeLogic.buyCondition = condition;
850       return;
851     }
852
853     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_23	/BitarenaToken (2).sol	L857 - L866	*Pending Fix*

/BitarenaToken (2).sol L857 - L866

```
856     ///   □ address of the Uniswap V3 Pool Created
857     function setUniswapV3Pool
858     (
859       address account
860     )
861     external
862     onlyOwner
863     {
864       mapAddressV3Pool[account] = true;
865       return;
866     }
867
868     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_25	/BitarenaToken (2).sol	L888 - L903	⚠ *Pending Fix*

/BitarenaToken (2).sol L888 - L903

```
887     ///   buy fee expressed as USD (fiat)
888     function setSwapFees
889     (
890       uint256 fee,
891       string memory feeType
892     )
893     external
894     onlyOwner
895     {
896       if (fee < 0 || fee > 250)
897         revert ErrorGeneric(fee, "BTA :: fee cannot be lower than 0 or higher than 2.5 USD
      (=250)");
898       if (feeType.equal("buy"))
899         instanceFeeLogic.numBuyFee = fee;
900       else if (feeType.equal("sell"))
901         instanceFeeLogic.numSellFee = fee;
902       return;
903     }
904
905     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_26	/BitarenaToken (2).sol	L909 - L918	⚠ *Pending Fix*

/BitarenaToken (2).sol L909 - L918

```
908     ///   □ flag to set
909     function setDebugging
910     (
911       bool _isDebugActive
912     )
913     external
914     onlyOwner
915     {
916       isDebugActive = _isDebugActive;
917       return;
918     }
919
920     // #endregion ➤ □ GETTER/SETTER
```

Issue Type

LONG NUMBER LITERALS

S. No.	Severity	Detection Method	Instances
G009	Gas	Automated	1

Description

Solidity supports multiple rational and integer literals, including decimal fractions and scientific notations. The use of very large numbers with too many digits was detected in the code that could have been optimized using a different n otation also supported by Solidity.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_71	/BitarenaToken (2).sol	L66 - L66	*Pending Fix*

/BitarenaToken (2).sol L66 - L66

```
65      ///   Token initial supply
66      uint256 public constant SUPPLY_TOTAL = 21000000;
67      /// @notice
68      ///   Current circulating supply BTA Token
```

Issue Type

MULTIPLICATION/DIVISION BY TWO SHOULD USE BIT SHIFTING

S. No.	Severity	Detection Method	Instances
G010	Gas	Automated	1

Description

`x * 2` is equal to `x << 1` and `x / 2` is equal to `x >> 1`. The MUL and DIV opcodes cost 5 gas, whereas SHL and SHR only cost 3 gas.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_35	/BitarenaToken (2).sol	L227 - L227	*Pending Fix*

/BitarenaToken (2).sol L227 - L227

```
226     _mint(instanceBetarenaAddresses.adrFoundingTeam,  (_onePercent * 5));
227     _mint(instanceBetarenaAddresses.adrAdvisoryBoard, (_onePercent * 2));
228     _mint(instanceBetarenaAddresses.adrInvestors,     (_onePercent * 4));
229     _mint(instanceBetarenaAddresses.adrTeam,          (_onePercent * 5));
```

Issue Type

NAMED RETURN OF LOCAL VARIABLE SAVES GAS AS COMPARED TO RETURN STATEMENT

S. No.	Severity	Detection Method	Instances
G011	Gas	Automated	1

Description

The function having a return type is found to be declaring a local variable for returning, which causes extra gas cons umption. This inefficiency arises because creating and manipulating local variables requires additional computational steps and memory allocation.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_6	/BitarenaToken (2).sol	L722 - L764	*Pending Fix*

/BitarenaToken (2).sol L722 - L764

```
721     ///   □ price of BTA token in USD
722     function calculateBitarenaPriceInStableCoinV2
723     (
724       address _adrBtaUsdtPool
725     )
726     private
727     nonReentrant
728     returns
729     (
730       uint256
731     )
732     {
733       (uint160 sqrtPriceX96, , , , , , ) = IUniswapV3Pool(_adrBtaUsdtPool).slot0();
734       address token0 = IUniswapV3Pool(_adrBtaUsdtPool).token0();
735       address token1 = IUniswapV3Pool(_adrBtaUsdtPool).token1();
736
737       // [□]
738       if (isDebugActive) emit DebugSwapSnapshot(sqrtPriceX96);
739
740       uint256 sqrtPriceX96Pow = 0;
741
742       // ┌────
```

/BitarenaToken (2).sol L722 - L764

```
741
742       // ┌──────
743       // │ NOTE:
744       // │ |: Check token ordering of V3Pool, which inflicts on the final price calculation.
745       // └──────
746       if (token0 == address(this))
747       {
748         sqrtPriceX96Pow = uint256(sqrtPriceX96 * 10**12);
749       }
750       else if (token1 == address(this))
751       {
752         sqrtPriceX96Pow = uint256(sqrtPriceX96);
753       }
754       else
755       {
756         return 0;
757       }
758
759       uint256 price = sqrtPriceX96Pow / 2**96;
760       price = price**2;
761       price = price * 10**6;
762
763       return price;
764     }
765
766     /// @notice
```

Issue Type

OPTIMIZING ADDRESS ID MAPPING

S. No.	Severity	Detection Method	Instances
G012	Gas	Automated	4

Description

Combining multiple address/ID mappings into a single mapping using a struct enhances storage efficiency, simplifies code, and reduces gas costs, resulting in a more streamlined and cost-effective smart contract design.
It saves storage slot for the mapping and depending on the circumstances and sizes of types, it can avoid a Gsset (2 0000 gas) per mapping combined. Reads and subsequent writes can also be cheaper when a function requires both values and they fit in the same storage slot.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_14	/BitarenaToken (2).sol	L72 - L72	*Pending Fix*

/BitarenaToken (2).sol L72 - L72

```
71      ///   Mapping for, general addresses that should not have imposed transaction fees.
72      mapping (address addressGeneralExcluded => bool isExcluded) private mapAddressExcluded;
73      /// @notice
74      ///   Mapping for, swap-related UniswapV3Pool (addresss) involved in BUY action logic.
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_15	/BitarenaToken (2).sol	L75 - L75	*Pending Fix*

/BitarenaToken (2).sol L75 - L75

```
74    ///   Mapping for, swap-related UniswapV3Pool (addresss) involved in BUY action logic.
75    mapping (address addressUniswapV3Excluded => bool isExcludedForBuy) private mapAdrExclud
      edForBuy;
76    /// @notice
77    ///   Mapping for, swap-related UniswapV3Pool (addresss) involved in SELL action logi
      c.
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_16	/BitarenaToken (2).sol	L78 - L78	*Pending Fix*

/BitarenaToken (2).sol L78 - L78

```
77    ///   Mapping for, swap-related UniswapV3Pool (addresss) involved in SELL action logi
      c.
78    mapping (address addressUniswapV3Excluded => bool isExcludedForSell) private mapAdrExclu
      dedForSell;
79    /// @notice
80    ///   Mapping ofor, Bitarena Token addresses of (UniswapV3) liquidity pools.
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_17	/BitarenaToken (2).sol	L81 - L81	⚠ *Pending Fix*

/BitarenaToken (2).sol L81 - L81

```
80      ///   Mapping ofor, Bitarena Token addresses of (UniswapV3) liquidity pools.
81      mapping (address addressV3Pool => bool isPool) private mapAddressV3Pool;
82      /// @notice
83      ///   Keeping track of current SwapContext
```

Issue Type

PUBLIC CONSTANTS CAN BE PRIVATE

S. No.	Severity	Detection Method	Instances
G013	Gas	Automated	1

Description

Public constant variables cost more gas because the EVM automatically creates getter functions for them and adds entries to the method ID table. The values can be read from the source code instead.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_81	/BitarenaToken (2).sol	L66 - L66	*Pending Fix*

/BitarenaToken (2).sol L66 - L66

```
65      ///   Token initial supply
66      uint256 public constant SUPPLY_TOTAL = 21000000;
67      /// @notice
68      ///   Current circulating supply BTA Token
```

Issue Type

STORAGE VARIABLE CACHING IN MEMORY

S. No.	Severity	Detection Method	Instances
G014	Gas	Automated	12

Description

The contract is using the state variable multiple times in the function.
`SLOADs` are expensive (100 gas after the 1st one) compared to `MLOAD` / `MSTORE` (3 gas each).

Bug ID	File Location	Line No.	Action Taken
SSP_62437_52	/BitarenaToken (2).sol	L170 - L255	*Pending Fix*

/BitarenaToken (2).sol L170 - L255

```
169     ///   BitarenaToken
170     constructor
171     (
172       string memory _name,
173       string memory _symbol,
174
175       address _adrFoundingTeam,
176       address _adrAdvisoryBoard,
177       address _adrInvestors,
178       address _adrTeam,
179       address _adrParticipants,
180       address _adrMarketing,
181       address _adrLiquidity,
182       address _adrReserve,
183
184       address _adrFeeAddress,
185       address _adrUniswapPermit2,
186       address _adrUniswapUniversalRouter
187     )
188     ERC20(_name, _symbol)
189     ERC20Permit(_name)
190     Ownable(msg.sender)
```

/BitarenaToken (2).sol L170 - L255

```
    ERC20Permit(_name)
    Ownable(msg.sender)
    {
      instanceBetarenaAddresses.adrFoundingTeam  = _adrFoundingTeam;
      instanceBetarenaAddresses.adrAdvisoryBoard = _adrAdvisoryBoard;
      instanceBetarenaAddresses.adrInvestors     = _adrInvestors;
      instanceBetarenaAddresses.adrTeam          = _adrTeam;
      instanceBetarenaAddresses.adrParticipants  = _adrParticipants;
      instanceBetarenaAddresses.adrMarketing     = _adrMarketing;
      instanceBetarenaAddresses.adrLiquidity     = _adrLiquidity;
      instanceBetarenaAddresses.adrReserve       = _adrReserve;

      instanceFeeLogic.adrFeeDeposit             = _adrFeeAddress;
      instanceFeeLogic.adrUniswapPermit2         = _adrUniswapPermit2;
      instanceFeeLogic.adrUniswapUniversalRouter = _adrUniswapUniversalRouter;

      // [□]
      // solhint-disable no-console
      // console.log(unicode"□ [checkpoint] :: initialize(..)");
      // console.log(unicode"□ [var] _adrFoundingTeam :: %s", _adrFoundingTeam);
      // console.log(unicode"□ [var] _adrAdvisoryBoard :: %s", _adrAdvisoryBoard);
      // console.log(unicode"□ [var] _adrInvestors :: %s", _adrInvestors);
      // console.log(unicode"□ [var] _adrTeam :: %s", _adrTeam);
      // console.log(unicode"□ [var] _adrParticipants :: %s", _adrParticipants);
      // console.log(unicode"□ [var] _adrMarketing :: %s", _adrMarketing);
      // console.log(unicode"□ [var] _adrLiquidity :: %s", _adrLiquidity);
      // console.log(unicode"□ [var] _adrReserve :: %s", _adrReserve);
      // console.log(unicode"□ [var] _adrUniswapPermit2 :: %s", _adrUniswapPermit2);
      // console.log(unicode"□ [var] address(this) :: %s", address(this));
      // solhint-enable no-console

      // ┌──────────────────────────────────────────────────────────────────────────────┐
      // │ □ │ MAIN INTIALIZER LOGIC                                                    │
      // └──────────────────────────────────────────────────────────────────────────────┘

      uint256 _onePercent = calcOnePercentOfTotalSupply();

      _mint(instanceBetarenaAddresses.adrFoundingTeam,  (_onePercent * 5));
      _mint(instanceBetarenaAddresses.adrAdvisoryBoard, (_onePercent * 2));
      _mint(instanceBetarenaAddresses.adrInvestors,     (_onePercent * 4));
      _mint(instanceBetarenaAddresses.adrTeam,          (_onePercent * 5));
      _mint(instanceBetarenaAddresses.adrParticipants,  (_onePercent * 10));
      _mint(instanceBetarenaAddresses.adrMarketing,     (_onePercent * 10));
      _mint(instanceBetarenaAddresses.adrLiquidity,     (_onePercent * 60));
```

/BitarenaToken (2).sol L170 - L255

```
231     _mint(instanceBetarenaAddresses.adrMarketing,     (_onePercent * 10));
232     _mint(instanceBetarenaAddresses.adrLiquidity,     (_onePercent * 60));
233     _mint(instanceBetarenaAddresses.adrReserve,       (_onePercent * 4));
234
235     // Causes Error:
236     // ProviderError: Error: VM Exception while processing transaction: reverted with panic code 0
    x11 (Arithmetic operation overflowed outside of an unchecked block)
237     updateCirculatingSupply();
238
239     mapAddressExcluded[owner()]                                  = true;
240     mapAddressExcluded[address(this)]                            = true;
241     mapAddressExcluded[instanceBetarenaAddresses.adrFoundingTeam]  = true;
242     mapAddressExcluded[instanceBetarenaAddresses.adrAdvisoryBoard] = true;
243     mapAddressExcluded[instanceBetarenaAddresses.adrInvestors]     = true;
244     mapAddressExcluded[instanceBetarenaAddresses.adrTeam]          = true;
245     mapAddressExcluded[instanceBetarenaAddresses.adrParticipants]  = true;
246     mapAddressExcluded[instanceBetarenaAddresses.adrMarketing]     = true;
247     mapAddressExcluded[instanceBetarenaAddresses.adrLiquidity]     = true;
248     mapAddressExcluded[instanceBetarenaAddresses.adrReserve]       = true;
249
250     instanceFeeLogic.numBuyFee  = 100;
251     instanceFeeLogic.numSellFee = 50;
252     instanceFeeLogic.buyCondition = 0;
253
254     return;
255   }
256
257   // #region ➤ □ METHODS
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_52	/BitarenaToken (2).sol	L170 - L255	*Pending Fix*

/BitarenaToken (2).sol

L170 - L255

```
169     ///   BitarenaToken
170     constructor
171     (
172       string memory _name,
173       string memory _symbol,
174
175       address _adrFoundingTeam,
176       address _adrAdvisoryBoard,
177       address _adrInvestors,
178       address _adrTeam,
179       address _adrParticipants,
180       address _adrMarketing,
181       address _adrLiquidity,
182       address _adrReserve,
183
184       address _adrFeeAddress,
185       address _adrUniswapPermit2,
186       address _adrUniswapUniversalRouter
187     )
188     ERC20(_name, _symbol)
189     ERC20Permit(_name)
190     Ownable(msg.sender)
191     {
```

/BitarenaToken (2).sol L170 - L255

```
    ERC20Permit(_name)
    Ownable(msg.sender)
    {
      instanceBetarenaAddresses.adrFoundingTeam  = _adrFoundingTeam;
      instanceBetarenaAddresses.adrAdvisoryBoard = _adrAdvisoryBoard;
      instanceBetarenaAddresses.adrInvestors     = _adrInvestors;
      instanceBetarenaAddresses.adrTeam          = _adrTeam;
      instanceBetarenaAddresses.adrParticipants  = _adrParticipants;
      instanceBetarenaAddresses.adrMarketing     = _adrMarketing;
      instanceBetarenaAddresses.adrLiquidity     = _adrLiquidity;
      instanceBetarenaAddresses.adrReserve       = _adrReserve;

      instanceFeeLogic.adrFeeDeposit             = _adrFeeAddress;
      instanceFeeLogic.adrUniswapPermit2         = _adrUniswapPermit2;
      instanceFeeLogic.adrUniswapUniversalRouter = _adrUniswapUniversalRouter;

      // [□]
      // solhint-disable no-console
      // console.log(unicode"□ [checkpoint] :: initialize(..)");
      // console.log(unicode"□ [var] _adrFoundingTeam :: %s", _adrFoundingTeam);
      // console.log(unicode"□ [var] _adrAdvisoryBoard :: %s", _adrAdvisoryBoard);
      // console.log(unicode"□ [var] _adrInvestors :: %s", _adrInvestors);
      // console.log(unicode"□ [var] _adrTeam :: %s", _adrTeam);
      // console.log(unicode"□ [var] _adrParticipants :: %s", _adrParticipants);
      // console.log(unicode"□ [var] _adrMarketing :: %s", _adrMarketing);
      // console.log(unicode"□ [var] _adrLiquidity :: %s", _adrLiquidity);
      // console.log(unicode"□ [var] _adrReserve :: %s", _adrReserve);
      // console.log(unicode"□ [var] _adrUniswapPermit2 :: %s", _adrUniswapPermit2);
      // console.log(unicode"□ [var] address(this) :: %s", address(this));
      // solhint-enable no-console

      // ┌──────────────────────────────────────────────────────────────────────────────┐
      // │ □ │ MAIN INTIALIZER LOGIC                                                    │
      // └──────────────────────────────────────────────────────────────────────────────┘

      uint256 _onePercent = calcOnePercentOfTotalSupply();

      _mint(instanceBetarenaAddresses.adrFoundingTeam,  (_onePercent * 5));
      _mint(instanceBetarenaAddresses.adrAdvisoryBoard, (_onePercent * 2));
      _mint(instanceBetarenaAddresses.adrInvestors,     (_onePercent * 4));
      _mint(instanceBetarenaAddresses.adrTeam,          (_onePercent * 5));
      _mint(instanceBetarenaAddresses.adrParticipants,  (_onePercent * 10));
      _mint(instanceBetarenaAddresses.adrMarketing,     (_onePercent * 10));
      _mint(instanceBetarenaAddresses.adrLiquidity,     (_onePercent * 60));
```

/BitarenaToken (2).sol L170 - L255

```
231     _mint(instanceBetarenaAddresses.adrMarketing,     (_onePercent * 10));
232     _mint(instanceBetarenaAddresses.adrLiquidity,     (_onePercent * 60));
233     _mint(instanceBetarenaAddresses.adrReserve,       (_onePercent * 4));
234
235     // Causes Error:
236     // ProviderError: Error: VM Exception while processing transaction: reverted with panic code 0
    x11 (Arithmetic operation overflowed outside of an unchecked block)
237     updateCirculatingSupply();
238
239     mapAddressExcluded[owner()]                                  = true;
240     mapAddressExcluded[address(this)]                            = true;
241     mapAddressExcluded[instanceBetarenaAddresses.adrFoundingTeam]  = true;
242     mapAddressExcluded[instanceBetarenaAddresses.adrAdvisoryBoard] = true;
243     mapAddressExcluded[instanceBetarenaAddresses.adrInvestors]     = true;
244     mapAddressExcluded[instanceBetarenaAddresses.adrTeam]          = true;
245     mapAddressExcluded[instanceBetarenaAddresses.adrParticipants]  = true;
246     mapAddressExcluded[instanceBetarenaAddresses.adrMarketing]     = true;
247     mapAddressExcluded[instanceBetarenaAddresses.adrLiquidity]     = true;
248     mapAddressExcluded[instanceBetarenaAddresses.adrReserve]       = true;
249
250     instanceFeeLogic.numBuyFee  = 100;
251     instanceFeeLogic.numSellFee = 50;
252     instanceFeeLogic.buyCondition = 0;
253
254     return;
255   }
256
257   // #region ➤ □ METHODS
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_53	/BitarenaToken (2).sol	L391 - L480	⚠ *Pending Fix*

/BitarenaToken (2).sol L391 - L480

```
390     ///   amount to transfer
391     function transferBuySellTakeFees
392     (
393       address sender,
394       address recipient,
395       uint256 amount
396     )
397     internal
398     {
399       // ┌──────
400       // │ CHECK |: Preliminary check (w/ exit)
401       // ├──────
402       // │ 1. Check if address is excluded from general fee
403       // │ 2. Check if address is 0x0 (a void)
404       // └──────
405       if
406       (
407         (
408           mapAddressExcluded[sender]
409           || mapAddressExcluded[recipient]
410         )
411         ||
412         (
```

/BitarenaToken (2).sol L391 - L480

```
        )
        ||
        (
          sender == address(0)
          || recipient == address(0)
        )
      )
      {
        // [□]
        if (isDebugActive) emit DebugTransactionStandard(sender, recipient);
        return;
      }

      // ╭──────
      // │ CHECK |: Buy-Action & Fee Gathering
      // ╰──────
      if (swapDetectType(sender, recipient, "buy"))
      {
        // [□]
        if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numBuyFee, "buy");

        uint256 buyFeeAmount;
        uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

        unchecked
        {
          buyFeeAmount = calculateBitarenaFee(instanceFeeLogic.numBuyFee, priceBtaFor1Usd);
          if (buyFeeAmount >= amount)
          {
            buyFeeAmount = 0;
            return;
          }
        }

        // ╭──────
        // │ NOTE:
        // │ ➤ 1XY% of the buyFeeAmount is taken
        // ╰──────
        super._update(recipient, instanceFeeLogic.adrFeeDeposit, buyFeeAmount);
      }

      // ╭──────
      // │ CHECK |: Sell-Action & Fee Gathering
      // ╰──────
```

/BitarenaToken (2).sol L391 - L480

```
    // │ CHECK |: Sell-Action & Fee Gathering
    // └──────
    if (swapDetectType(sender, recipient, "sell"))
    {
      // [⌷]
      if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numSellFee, "sell");

      uint256 sellFeeAmount;
      uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

      unchecked
      {
        sellFeeAmount = calculateBitarenaFee(instanceFeeLogic.numSellFee, priceBtaFor1Usd);
        if (amount + sellFeeAmount >= balanceOf(sender))
        {
          sellFeeAmount = 0;
          return;
        }
      }

      // ┌──────
      // │ NOTE:
      // │ ➤ 1XY% of the sellFeeAmount is taken
      // └──────
      super._update(sender, instanceFeeLogic.adrFeeDeposit, sellFeeAmount);
    }

    return;
  }

  /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_53	/BitarenaToken (2).sol	L391 - L480	⚠ *Pending Fix*

/BitarenaToken (2).sol L391 - L480

```
390     ///   amount to transfer
391     function transferBuySellTakeFees
392     (
393       address sender,
394       address recipient,
395       uint256 amount
396     )
397     internal
398     {
399       // ┌──────
400       // │ CHECK |: Preliminary check (w/ exit)
401       // ├──────
402       // │ 1. Check if address is excluded from general fee
403       // │ 2. Check if address is 0x0 (a void)
404       // └──────
405       if
406       (
407         (
408           mapAddressExcluded[sender]
409           || mapAddressExcluded[recipient]
410         )
411         ||
412         (
```

/BitarenaToken (2).sol L391 - L480

```
        )
        ||
        (
          sender == address(0)
          || recipient == address(0)
        )
      )
      {
        // [□]
        if (isDebugActive) emit DebugTransactionStandard(sender, recipient);
        return;
      }

      // ╭──────
      // │ CHECK |: Buy-Action & Fee Gathering
      // ╰──────
      if (swapDetectType(sender, recipient, "buy"))
      {
        // [□]
        if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numBuyFee, "buy");

        uint256 buyFeeAmount;
        uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

        unchecked
        {
          buyFeeAmount = calculateBitarenaFee(instanceFeeLogic.numBuyFee, priceBtaFor1Usd);
          if (buyFeeAmount >= amount)
          {
            buyFeeAmount = 0;
            return;
          }
        }

        // ╭──────
        // │ NOTE:
        // │ ➤ 1XY% of the buyFeeAmount is taken
        // ╰──────
        super._update(recipient, instanceFeeLogic.adrFeeDeposit, buyFeeAmount);
      }

      // ╭──────
      // │ CHECK |: Sell-Action & Fee Gathering
      // ╰──────
```

/BitarenaToken (2).sol L391 - L480

```
    // │ CHECK |: Sell-Action & Fee Gathering
    // └──────
    if (swapDetectType(sender, recipient, "sell"))
    {
      // [⌷]
      if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numSellFee, "sell");

      uint256 sellFeeAmount;
      uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

      unchecked
      {
        sellFeeAmount = calculateBitarenaFee(instanceFeeLogic.numSellFee, priceBtaFor1Usd);
        if (amount + sellFeeAmount >= balanceOf(sender))
        {
          sellFeeAmount = 0;
          return;
        }
      }

      // ┌──────
      // │ NOTE:
      // │ ➤ 1XY% of the sellFeeAmount is taken
      // └──────
      super._update(sender, instanceFeeLogic.adrFeeDeposit, sellFeeAmount);
    }

    return;
  }

  /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_53	/BitarenaToken (2).sol	L391 - L480	⚠ *Pending Fix*

/BitarenaToken (2).sol L391 - L480

```
390     ///   amount to transfer
391     function transferBuySellTakeFees
392     (
393       address sender,
394       address recipient,
395       uint256 amount
396     )
397     internal
398     {
399       // ┌──────
400       // │ CHECK |: Preliminary check (w/ exit)
401       // ├──────
402       // │ 1. Check if address is excluded from general fee
403       // │ 2. Check if address is 0x0 (a void)
404       // └──────
405       if
406       (
407         (
408           mapAddressExcluded[sender]
409           || mapAddressExcluded[recipient]
410         )
411         ||
412         (
```

/BitarenaToken (2).sol L391 - L480

```
        )
        ||
        (
          sender == address(0)
          || recipient == address(0)
        )
      )
      {
        // [⌐]
        if (isDebugActive) emit DebugTransactionStandard(sender, recipient);
        return;
      }

      // ┌──────
      // │ CHECK |: Buy-Action & Fee Gathering
      // └──────
      if (swapDetectType(sender, recipient, "buy"))
      {
        // [⌐]
        if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numBuyFee, "buy");

        uint256 buyFeeAmount;
        uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

        unchecked
        {
          buyFeeAmount = calculateBitarenaFee(instanceFeeLogic.numBuyFee, priceBtaFor1Usd);
          if (buyFeeAmount >= amount)
          {
            buyFeeAmount = 0;
            return;
          }
        }

        // ┌──────
        // │ NOTE:
        // │ ➤ 1XY% of the buyFeeAmount is taken
        // └──────
        super._update(recipient, instanceFeeLogic.adrFeeDeposit, buyFeeAmount);
      }

      // ┌──────
      // │ CHECK |: Sell-Action & Fee Gathering
      // └──────
```

/BitarenaToken (2).sol L391 - L480

```
    // │ CHECK |: Sell-Action & Fee Gathering
    // └──────
    if (swapDetectType(sender, recipient, "sell"))
    {
      // [⌷]
      if (isDebugActive) emit DebugTransactionSwap(sender, recipient, instanceFeeLogic.numSellFee, "sell");

      uint256 sellFeeAmount;
      uint256 priceBtaFor1Usd = calculateBitarenaPriceInStableCoinV2(instanceFeeLogic.adrBtaUsdPool);

      unchecked
      {
        sellFeeAmount = calculateBitarenaFee(instanceFeeLogic.numSellFee, priceBtaFor1Usd);
        if (amount + sellFeeAmount >= balanceOf(sender))
        {
          sellFeeAmount = 0;
          return;
        }
      }

      // ┌──────
      // │ NOTE:
      // │ ➤ 1XY% of the sellFeeAmount is taken
      // └──────
      super._update(sender, instanceFeeLogic.adrFeeDeposit, sellFeeAmount);
    }

    return;
  }

  /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_54	/BitarenaToken (2).sol	L492 - L599	⚠ *Pending Fix*

/BitarenaToken (2).sol L492 - L599

```
491     ///   □ 'true' IF swap type is detected
492     function swapDetectType
493     (
494       address sender,
495       address recipient,
496       string memory swapType
497     )
498     internal
499     returns
500     (
501       bool
502     )
503     {
504
505       // ┌──────────────────────────────────────────────────────────────────────────────────────────────────────────────┐
506       // │ □ │ BUY                                                                                                          │
507       // ├──────────────────────────────────────────────────────────────────────────────────────────────────────────────┤
508       // │ CONDTION [0]                                                                                                 │
509       // │  Description                                                                                                 │
510       // │  :: Pre-Swap Detection                                                                                       │
511       // ├──────────────────────────────────────────────────────────────────────────────────────────────────────────────┤
512       // │ 1. 'UniversalRouter' = is 'msg.sender'                                                                       │
513       // │ 2. 'UniversalRouter' = is 'sender/from'                                                                      │
```

/BitarenaToken (2).sol L492 - L599

```
    // ├──────────────────────────────────────────────────────────────────────┤
    // │ 1. 'UniversalRouter' = is 'msg.sender'                                │
    // │ 2. 'UniversalRouter' = is 'sender/from'                               │
    // │ 3. 'user'            = is 'recipient/to'                              │
    // ├──────────────────────────────────────────────────────────────────────┤
    // │ CONDTION [1]                                                          │
    // │  Description                                                          │
    // │  :: Post-Swap Detection                                               │
    // │  :: Typically combined with CONDTION [0], but can be a standalone     │
    // │  :: **WORKING** condition                                             │
    // ├──────────────────────────────────────────────────────────────────────┤
    // │ 1. 'PancakeV3Pool|UniswapV3Pool' = is 'msg.sender'                    │
    // │ 2. 'PancakeV3Pool|UniswapV3Pool' = is 'sender/from'                   │
    // │ 3. 'user'                        = is 'recipient/to'                  │
    // │ 4. 'user'                        = is 'tx.origin'                     │
    // └──────────────────────────────────────────────────────────────────────┘

    if
    (
      swapType.equal("buy")
      && instanceFeeLogic.numBuyFee != 0
      && instanceFeeLogic.buyCondition == 0
    )
    {
      string memory mapBuyKey = string.concat(Strings.toString(block.number), "_", Strings.toString(block.timestamp), "_", Strings.toString(tx.gasprice));

      // [□]
      if (isDebugActive) emit DebugTransactionSwapCheckpoint(mapBuyKey);

      // ┌──────
      // │ NOTE:
      // │ |: Validating the (pre-swap) 'Buy' condition
      // └──────
      if
      (
        msg.sender == sender
        && isUniswapV3Pool(sender)
        && mapSwapContext[tx.origin][mapBuyKey] == address(0)
      )
      {
        // [□]
        if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-0]");

        mapSwapContext[tx.origin][mapBuyKey] = sender;
```

/BitarenaToken (2).sol L492 - L599

```
553
554         mapSwapContext[tx.origin][mapBuyKey] = sender;
555         return false;
556       }
557       // ╭──────
558       // │ NOTE:
559       // │ |: Validating the (post-swap) 'Buy' condition
560       // ╰──────
561       else if
562       (
563         tx.origin == recipient
564         && msg.sender == instanceFeeLogic.adrUniswapUniversalRouter
565         && sender == instanceFeeLogic.adrUniswapUniversalRouter
566         && mapSwapContext[tx.origin][mapBuyKey] != address(0)
567         && !mapAdrExcludedForBuy[(mapSwapContext[tx.origin][mapBuyKey])]
568       )
569       {
570         // [□]
571         if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-1]");
572
573         delete mapSwapContext[tx.origin][mapBuyKey];
574         return true;
575       }
576
577       return false;
578     }
579
580     // ╭──────────────────────────────────────────────────────────────────────────────────╮
581     // │ □ | SELL                                                                         |
582     // ├──────────────────────────────────────────────────────────────────────────────────┤
583     // | CONDTION [1]                                                                     |
584     // | 1. 'Permit2'                    = is 'msg.sender'                                |
585     // | 2. 'user'                       = is 'sender/from'                               |
586     // | 3. 'PancakeV3Pool|UniswapV3Pool' = is 'recipient/to'                            |
587     // ╰──────────────────────────────────────────────────────────────────────────────────╯
588
589     if
590     (
591       swapType.equal("sell")
592       && instanceFeeLogic.numSellFee != 0
593       && msg.sender == instanceFeeLogic.adrUniswapPermit2
594       && !isUniswapV3Pool(sender)
595       && !mapAdrExcludedForSell[recipient]
596     ) return true;
```

/BitarenaToken (2).sol L492 - L599

```
595         && !mapAdrExcludedForSell[recipient]
596       ) return true;
597
598       return false;
599     }
600
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_54	/BitarenaToken (2).sol	L492 - L599	⚠️ *Pending Fix*

/BitarenaToken (2).sol L492 - L599

```
491     ///   □ 'true' IF swap type is detected
492     function swapDetectType
493     (
494       address sender,
495       address recipient,
496       string memory swapType
497     )
498     internal
499     returns
500     (
501       bool
502     )
503     {
504
505       // ┌──────────────
```

/BitarenaToken (2).sol L492 - L599

```
    // ├──────────────────────────────────────────────────────────────────────┤
    // │ 1. 'UniversalRouter' = is 'msg.sender'                                │
    // │ 2. 'UniversalRouter' = is 'sender/from'                               │
    // │ 3. 'user'            = is 'recipient/to'                              │
    // ├──────────────────────────────────────────────────────────────────────┤
    // │ CONDTION [1]                                                          │
    // │  Description                                                          │
    // │  :: Post-Swap Detection                                               │
    // │  :: Typically combined with CONDTION [0], but can be a standalone     │
    // │  :: **WORKING** condition                                             │
    // ├──────────────────────────────────────────────────────────────────────┤
    // │ 1. 'PancakeV3Pool|UniswapV3Pool' = is 'msg.sender'                    │
    // │ 2. 'PancakeV3Pool|UniswapV3Pool' = is 'sender/from'                   │
    // │ 3. 'user'                        = is 'recipient/to'                  │
    // │ 4. 'user'                        = is 'tx.origin'                     │
    // └──────────────────────────────────────────────────────────────────────┘

    if
    (
      swapType.equal("buy")
      && instanceFeeLogic.numBuyFee != 0
      && instanceFeeLogic.buyCondition == 0
    )
    {
      string memory mapBuyKey = string.concat(Strings.toString(block.number), "_", Strings.toString(block.timestamp), "_", Strings.toString(tx.gasprice));

      // [□]
      if (isDebugActive) emit DebugTransactionSwapCheckpoint(mapBuyKey);

      // ┌──────
      // │ NOTE:
      // │ |: Validating the (pre-swap) 'Buy' condition
      // └──────
      if
      (
        msg.sender == sender
        && isUniswapV3Pool(sender)
        && mapSwapContext[tx.origin][mapBuyKey] == address(0)
      )
      {
        // [□]
        if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-0]");

        mapSwapContext[tx.origin][mapBuyKey] = sender;
```

/BitarenaToken (2).sol L492 - L599

```
553
554         mapSwapContext[tx.origin][mapBuyKey] = sender;
555         return false;
556       }
557       // ╭──────
558       // │ NOTE:
559       // │ |: Validating the (post-swap) 'Buy' condition
560       // ╰──────
561       else if
562       (
563         tx.origin == recipient
564         && msg.sender == instanceFeeLogic.adrUniswapUniversalRouter
565         && sender == instanceFeeLogic.adrUniswapUniversalRouter
566         && mapSwapContext[tx.origin][mapBuyKey] != address(0)
567         && !mapAdrExcludedForBuy[(mapSwapContext[tx.origin][mapBuyKey])]
568       )
569       {
570         // [□]
571         if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-1]");
572
573         delete mapSwapContext[tx.origin][mapBuyKey];
574         return true;
575       }
576
577       return false;
578     }
579
580     // ╭──────────────────────────────────────────────────────────────────────────────────╮
581     // │ □ | SELL                                                                         |
582     // ├──────────────────────────────────────────────────────────────────────────────────┤
583     // | CONDTION [1]                                                                     |
584     // | 1. 'Permit2'                    = is 'msg.sender'                                |
585     // | 2. 'user'                       = is 'sender/from'                               |
586     // | 3. 'PancakeV3Pool|UniswapV3Pool' = is 'recipient/to'                            |
587     // ╰──────────────────────────────────────────────────────────────────────────────────╯
588
589     if
590     (
591       swapType.equal("sell")
592       && instanceFeeLogic.numSellFee != 0
593       && msg.sender == instanceFeeLogic.adrUniswapPermit2
594       && !isUniswapV3Pool(sender)
595       && !mapAdrExcludedForSell[recipient]
596     ) return true;
```

/BitarenaToken (2).sol L492 - L599

```
        && !mapAdrExcludedForSell[recipient]
      ) return true;

      return false;
    }

```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_54	/BitarenaToken (2).sol	L492 - L599	⚠️ *Pending Fix*

/BitarenaToken (2).sol L492 - L599

```
    ///   □ 'true' IF swap type is detected
    function swapDetectType
    (
      address sender,
      address recipient,
      string memory swapType
    )
    internal
    returns
    (
      bool
    )
    {

      // ┌──────────────────────────────────────────────────────────────────────────────────────────────────────┐
      // │ □ │ BUY                                                                                                  │
      // ├──────────────────────────────────────────────────────────────────────────────────────────────────────┤
      // │ CONDTION [0]                                                                                         │
      // │  Description                                                                                         │
      // │  :: Pre-Swap Detection                                                                               │
      // ├──────────────────────────────────────────────────────────────────────────────────────────────────────┤
      // │ 1. 'UniversalRouter' = is 'msg.sender'                                                               │
      // │ 2. 'UniversalRouter' = is 'sender/from'                                                              │
```

/BitarenaToken (2).sol L492 - L599

```
    // ├──────────────────────────────────────────────────────────────────────────┤
    // │ 1. 'UniversalRouter' = is 'msg.sender'                                   │
    // │ 2. 'UniversalRouter' = is 'sender/from'                                  │
    // │ 3. 'user'            = is 'recipient/to'                                 │
    // ├──────────────────────────────────────────────────────────────────────────┤
    // │ CONDTION [1]                                                             │
    // │  Description                                                             │
    // │  :: Post-Swap Detection                                                  │
    // │  :: Typically combined with CONDTION [0], but can be a standalone        │
    // │  :: **WORKING** condition                                                │
    // ├──────────────────────────────────────────────────────────────────────────┤
    // │ 1. 'PancakeV3Pool|UniswapV3Pool' = is 'msg.sender'                       │
    // │ 2. 'PancakeV3Pool|UniswapV3Pool' = is 'sender/from'                      │
    // │ 3. 'user'                        = is 'recipient/to'                     │
    // │ 4. 'user'                        = is 'tx.origin'                        │
    // └──────────────────────────────────────────────────────────────────────────┘

    if
    (
      swapType.equal("buy")
      && instanceFeeLogic.numBuyFee != 0
      && instanceFeeLogic.buyCondition == 0
    )
    {
      string memory mapBuyKey = string.concat(Strings.toString(block.number), "_", Strings.toString(block.timestamp), "_", Strings.toString(tx.gasprice));

      // [□]
      if (isDebugActive) emit DebugTransactionSwapCheckpoint(mapBuyKey);

      // ┌──────
      // │ NOTE:
      // │ |: Validating the (pre-swap) 'Buy' condition
      // └──────
      if
      (
        msg.sender == sender
        && isUniswapV3Pool(sender)
        && mapSwapContext[tx.origin][mapBuyKey] == address(0)
      )
      {
        // [□]
        if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-0]");

        mapSwapContext[tx.origin][mapBuyKey] = sender;
```

/BitarenaToken (2).sol L492 - L599

```
553
554         mapSwapContext[tx.origin][mapBuyKey] = sender;
555         return false;
556       }
557       // ╭──────
558       // │ NOTE:
559       // │ |: Validating the (post-swap) 'Buy' condition
560       // ╰──────
561       else if
562       (
563         tx.origin == recipient
564         && msg.sender == instanceFeeLogic.adrUniswapUniversalRouter
565         && sender == instanceFeeLogic.adrUniswapUniversalRouter
566         && mapSwapContext[tx.origin][mapBuyKey] != address(0)
567         && !mapAdrExcludedForBuy[(mapSwapContext[tx.origin][mapBuyKey])]
568       )
569       {
570         // [□]
571         if (isDebugActive) emit DebugTransactionSwapCheckpoint("[checkpoint] :: Buy Condition [Step-1]");
572
573         delete mapSwapContext[tx.origin][mapBuyKey];
574         return true;
575       }
576
577       return false;
578     }
579
580     // ╭──────────────────────────────────────────────────────────────────────────────────╮
581     // │ □ | SELL                                                                         │
582     // ├──────────────────────────────────────────────────────────────────────────────────┤
583     // │ CONDTION [1]                                                                     │
584     // │ 1. 'Permit2'                     = is 'msg.sender'                               │
585     // │ 2. 'user'                        = is 'sender/from'                              │
586     // │ 3. 'PancakeV3Pool|UniswapV3Pool' = is 'recipient/to'                             │
587     // ╰──────────────────────────────────────────────────────────────────────────────────╯
588
589     if
590     (
591       swapType.equal("sell")
592       && instanceFeeLogic.numSellFee != 0
593       && msg.sender == instanceFeeLogic.adrUniswapPermit2
594       && !isUniswapV3Pool(sender)
595       && !mapAdrExcludedForSell[recipient]
596     ) return true;
```

/BitarenaToken (2).sol L492 - L599

```
595         && !mapAdrExcludedForSell[recipient]
596       ) return true;
597
598       return false;
599     }
600
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_55	/BitarenaToken (2).sol	L619 - L672	⚠️ *Pending Fix*

/BitarenaToken (2).sol L619 - L672

```
618     ///   |: based on the current balance of the non-circulating addresses
619     function updateCirculatingSupply()
620     private
621     {
622       if (totalSupply() == 0)
623       {
624         numCirculatingSupply = 0;
625         return;
626       }
627
628       // [□]
629       // solhint-disable no-console
630       // console.log(unicode"□ [var] numCirculatingSupply :: %s", numCirculatingSupply);
631       // console.log(unicode"□ [var] totalSupply() :: %s", totalSupply());
632       // solhint-enable no-console
633
634       uint256 numNonCirculatingSupply;
635       uint256 numTotalSupply = totalSupply();
636
637       // ┌──────
638       // │ CHECK: |:| for same wallet address (used in testing)
639       // └──────
640       if (instanceBetarenaAddresses.adrFoundingTeam == instanceBetarenaAddresses.adrAdvisory
      Board)
```

/BitarenaToken (2).sol L619 - L672

```
    // │ CHECK: |:| for same wallet address (used in testing)
    // └──────
    if (instanceBetarenaAddresses.adrFoundingTeam == instanceBetarenaAddresses.adrAdvisoryBoard)
    {
      numNonCirculatingSupply = balanceOf(instanceBetarenaAddresses.adrFoundingTeam);
    }
    else
    {
      numNonCirculatingSupply = balanceOf(instanceBetarenaAddresses.adrFoundingTeam);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrAdvisoryBoard);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrInvestors);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrTeam);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrParticipants);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrMarketing);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrLiquidity);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrReserve);
    }

    if (numTotalSupply < numNonCirculatingSupply)
    {
      numCirculatingSupply = 0;
    }
    else
    {
      numCirculatingSupply = numTotalSupply - numNonCirculatingSupply;
    }

    // [□]
    // solhint-disable no-console
    // console.log(unicode"□ [var] numNonCirculatingSupply :: %s", numNonCirculatingSupply);
    // console.log(unicode"□ [var] numCirculatingSupply :: %s", numCirculatingSupply);
    // solhint-enable no-console

    return;
  }

  /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_55	/BitarenaToken (2).sol	L619 - L672	⚠ *Pending Fix*

/BitarenaToken (2).sol L619 - L672

```
618     ///   |: based on the current balance of the non-circulating addresses
619     function updateCirculatingSupply()
620     private
621     {
622       if (totalSupply() == 0)
623       {
624         numCirculatingSupply = 0;
625         return;
626       }
627
628       // [□]
629       // solhint-disable no-console
630       // console.log(unicode"□ [var] numCirculatingSupply :: %s", numCirculatingSupply);
631       // console.log(unicode"□ [var] totalSupply() :: %s", totalSupply());
632       // solhint-enable no-console
633
634       uint256 numNonCirculatingSupply;
635       uint256 numTotalSupply = totalSupply();
636
637       // ┌──────
638       // │ CHECK: |:| for same wallet address (used in testing)
639       // └──────
640       if (instanceBetarenaAddresses.adrFoundingTeam == instanceBetarenaAddresses.adrAdvisory
      Board)
```

/BitarenaToken (2).sol L619 - L672

```
    // │ CHECK: |:| for same wallet address (used in testing)
    // └──────
    if (instanceBetarenaAddresses.adrFoundingTeam == instanceBetarenaAddresses.adrAdvisoryBoard)
    {
      numNonCirculatingSupply = balanceOf(instanceBetarenaAddresses.adrFoundingTeam);
    }
    else
    {
      numNonCirculatingSupply = balanceOf(instanceBetarenaAddresses.adrFoundingTeam);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrAdvisoryBoard);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrInvestors);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrTeam);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrParticipants);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrMarketing);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrLiquidity);
      numNonCirculatingSupply += balanceOf(instanceBetarenaAddresses.adrReserve);
    }

    if (numTotalSupply < numNonCirculatingSupply)
    {
      numCirculatingSupply = 0;
    }
    else
    {
      numCirculatingSupply = numTotalSupply - numNonCirculatingSupply;
    }

    // [□]
    // solhint-disable no-console
    // console.log(unicode"□ [var] numNonCirculatingSupply :: %s", numNonCirculatingSupply);
    // console.log(unicode"□ [var] numCirculatingSupply :: %s", numCirculatingSupply);
    // solhint-enable no-console

    return;
  }

  /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_56	/BitarenaToken (2).sol	L825 - L836	⚠ *Pending Fix*

/BitarenaToken (2).sol L825 - L836

```
824     ///   address of UiswapUniversalRouter
825     function setUniswapOfficialAddress
826     (
827       address _adrUniswapPermit2,
828       address _adrUniswapUniversalRouter
829     )
830     external
831     onlyOwner
832     {
833       instanceFeeLogic.adrUniswapPermit2 = _adrUniswapPermit2;
834       instanceFeeLogic.adrUniswapUniversalRouter = _adrUniswapUniversalRouter;
835       return;
836     }
837
838     /// @notice
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_57	/BitarenaToken (2).sol	L888 - L903	⚠ *Pending Fix*

/BitarenaToken (2).sol L888 - L903

```
887     ///   buy fee expressed as USD (fiat)
888     function setSwapFees
889     (
890       uint256 fee,
891       string memory feeType
892     )
893     external
894     onlyOwner
895     {
896       if (fee < 0 || fee > 250)
897         revert ErrorGeneric(fee, "BTA :: fee cannot be lower than 0 or higher than 2.5 USD
      (=250)");
898       if (feeType.equal("buy"))
899         instanceFeeLogic.numBuyFee = fee;
900       else if (feeType.equal("sell"))
901         instanceFeeLogic.numSellFee = fee;
902       return;
903     }
904
905     /// @notice
```

Issue Type

SUPERFLUOUS EVENT FIELDS

S. No.	Severity	Detection Method	Instances
G015	Gas	Automated	1

Description

`block.timestamp` and `block.number` are by default added to event information. Adding them manually costs ex tra gas.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_82	/BitarenaToken (2).sol	L355 - L355	*Pending Fix*

/BitarenaToken (2).sol L355 - L355

```
354         emit DebugTransaction(from, to, value);
355         emit DebugGlobalContextBlock(block.number, block.timestamp, block.chainid);
356         emit DebugGlobalContextMsgAndTx(msg.data, msg.sender, msg.value, tx.origin, tx.gaspr
      ice, _msgSender(), _msgData());
357       }
```

Issue Type

USE SELFBALANCE() INSTEAD OF ADDRESS(THIS).BALANCE

S. No.	Severity	Detection Method	Instances
G016	Gas	Automated	2

Description

In Solidity, efficient use of gas is paramount to ensure cost-effective execution on the Ethereum blockchain. Gas can be optimized when obtaining contract balance by using `selfbalance()` rather than `address(this).balance` because it bypasses gas costs and refunds, which are not required for obtaining the contract's balance.

Bug ID	File Location	Line No.	Action Taken
SSP_62437_29	/BitarenaToken (2).sol	L285 - L285	*Pending Fix*

/BitarenaToken (2).sol L285 - L285

```
284     {
285       return address(this).balance;
286     }
287
```

Bug ID	File Location	Line No.	Action Taken
SSP_62437_30	/BitarenaToken (2).sol	L296 - L296	*Pending Fix*

/BitarenaToken (2).sol L296 - L296

```
295
296         uint256 amount = address(this).balance;
297         address payable recipient = payable(owner());
298         (bool success,) = recipient.call{ value: amount }("BTA :: Amount Withdrawn from smart contract");
```

05. Scan History

● Critical ● High ● Medium ● Low ● Informational ● Gas

No	Date	Security Score	Scan Overview
1.	2025-02-12	87.61	● 0 ● 1 ● 3 ● 11 ● 23 ● 44

06. Disclaimer

The Reports neither endorse nor condemn any specific project or team, nor do they guarantee the security of any specific project. The contents of this report do not, and should not be interpreted as having any bearing on, the economics of tokens, token sales, or any other goods, services, or assets.

The security audit is not meant to replace functional testing done before a software release.

There is no warranty that all possible security issues of a particular smart contract(s) will be found by the tool, i.e., It is not guaranteed that there will not be any further findings based solely on the results of this evaluation.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. There is no warranty or representation made by this report to any Third Party in regards to the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business.

In no way should a third party use these reports to make any decisions about buying or selling a token, product, service, or any other asset. It should be noted that this report is not investment advice, is not intended to be relied on as investment advice, and has no endorsement of this project or team. It does not serve as a guarantee as to the project's absolute security.

The assessment provided by SolidityScan 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. SolidityScan owes no duty to any third party by virtue of publishing these Reports.

As one audit-based assessment cannot be considered comprehensive, we always recommend proceeding with several independent manual audits including manual audit and a public bug bounty program to ensure the security of the smart contracts.