CERTIK

Security Assessment

MarsEcosystem

Aug 20th, 2021

Table of Contents

Summary

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

The auditing process pays special attention to the following considerations:

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

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

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

Overview

Project Summary

Project Name	MarsEcosystem
Platform	BSC
Language	Solidity
Codebase	https://github.com/MarsEcosystem/mars-ecosystem
Commit	<8f72334a5a64e74a92de362560ab5639c4df4107 >

Audit Summary

Delivery Date	Aug 20, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	ⓘ Pending	⊗ Declined	ⓘ Acknowledged	Partially Resolved	✓ Resolved
● Critical	0	0	0	0	0	0
● Major	2	0	0	0	1	1
● Medium	1	0	0	0	1	0
● Minor	25	0	0	5	0	20
● Informational	8	0	0	1	0	7
● Discussion	0	0	0	0	0	0

Audit Scope

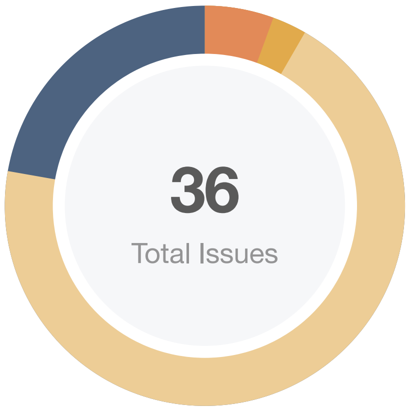
ID	File	SHA256 Checksum
USD	projects/contracts/base/USDMToken.sol	082f368a70558343c1891ef6a5680de0aa6e44add96d88de91bcf6528c70b7be
XMS	projects/contracts/base/XMSToken.sol	d797ad05191905af59382b2b4e7de154b961e99a520832b8a0615736b22bcd24
BNB	projects/contracts/bondingcurve/BNBBondingLCurve.sol	8d66b89510cc7b5b9cbff5c9257127e9cc8f14e2a163c512e5667a77b47a1e10
BUS	projects/contracts/bondingcurve/BUSDBondingLCurve.sol	6cfcc8851907d456363d4f478c21d1f913553de8caef0e0f94f79fb6fae98e9b
BLC	projects/contracts/bondingcurve/BondingLCurve.sol	13bf711204db6412976b018074b3193094ce824cd02ecbd4bcb05d642c01369d
CME	projects/contracts/core/Core.sol	3a502f9d545b3ea7ae97125a37aaf8b3bf10e5535840ba6c04c4fa22808d248f
PME	projects/contracts/core/Permissions.sol	6b626d92eb0c802a0d4ea423902af44e7c12fcba26ff35a46f2eb97cddcb0d9b
GAM	projects/contracts/dao/GovernorAlpha.sol	07dd14959afa5f114fa67f1b7be3fe6c4b5e96a09a5c3c1d32ee4a53fb9229cb
LTT	projects/contracts/dao/LinearTokenTimelockDelegator.sol	3ac15b58a8527a96d76bc98320ded5de30e5aed04d4e4b49777db404196a053b
PTT	projects/contracts/dao/PeriodTokenTimelockDelegator.sol	09424dd9840dbae0dd32a72ecd07faf12add1059e4ba0f48a2786b882a658cd9
STT	projects/contracts/dao/StraightTokenTimelockDelegator.sol	84122c28d615eb5bda879025295b0e38483b287598ec1fed1fb6515dc50e3b7e
TME	projects/contracts/dao/Timelock.sol	5f5c6cb101b3cbc7a32cb4591999ca17f662604610dfe8aeb245f28ad3242f17
BUD	projects/contracts/genesis/BUSDGenesisGroup.sol	64dfb47b9d8d67b50859fb88a9b55e84e4ed821afff0b68a2248e9f45127431f
IDO	projects/contracts/genesis/IDO.sol	404acb02a9077e90a901f13b0b19e7314b2811695da7de3ee47b621133d9370d
IMO	projects/contracts/genesis/IMO.sol	b467af3ce2e496ab1af63b59eafabe72674a8e25df1f6606de74fe8e2c6f1b63

ID	File	SHA256 Checksum
IME	projects/contracts/genesis/IMOExt.sol	b7990578ec1008d4c6d59e93bbd7d3bd8f47938002394119dec2f12c38f6bd22
LMM	projects/contracts/liquidity/LiquidityMiningMaster.sol	603a861c68893873f774a3647855108b51889d5df1a6c9f6a8cbeeaadb6460ea
MSR	projects/contracts/liquidity/MarsSwapRouter.sol	84d5983c481d420d44874bf5bcad4e867395fb0f50c20c6bafc696d00ebf66c6
BNL	projects/contracts/oracle/BNBLastPriceOracle.sol	dab77f744f7ac2f961e4c1a14e4359e8a4c62c04ce3743071d14b3a8a595e224
BUL	projects/contracts/oracle/BUSDLastPriceOracle.sol	cd62da3d8204e559b8d5375d81b6c1baccdbd32905895dcb1c17938b80d0d18a
COM	projects/contracts/oracle/CombinationOracle.sol	fc7668ec55fecb660625ac75bed16f21b6ea429312aa3c3eebbe6fe143bc6ecc
MSP	projects/contracts/oracle/MarsSwapPairCombOracle.sol	4fab4eb1649183d383859aa128ba301e38c9d4d2800bd35dc2a2157f86415dbf
OIM	projects/contracts/oracle/OracleIncentives.sol	9de7f81af3319ffcb3a9b8fa59fb144c101ec83151b72242fb21498fb62e03de
SMO	projects/contracts/oracle/SwapMiningOracle.sol	8b7f2626c5a4658ef9815a47e4db0afc251325d8dc0ecc531a02467dcd9c9156
BNV	projects/contracts/pcv/BNBVenusPCVDeposit.sol	196b226d86c2feabec5302e2cfcf59fb5014b214e96da0c3f7609c989f439db0
BUU	projects/contracts/pcv/BUSDUniswapPCVController.sol	d9448b07a8f637269ae3eee777321d79ed36d0574bfc8631d9f826ed9c10155a
BUP	projects/contracts/pcv/BUSDUniswapPCVDeposit.sol	8cf0ad0ecb6a49b2a8c4f3db751e32118a87df8e1789e3da1edca733727d7d4c
PCV	projects/contracts/pcv/PCVController.sol	37ee08900997d7e6367ac422787d162bd7e65a9134afd16be491e99797df91b2
PCS	projects/contracts/pcv/PCVSplitter.sol	71354045a7f3044d01056190d8cb6ab5f4e26a0bf288df56acc7e0d3f782e52b
PCU	projects/contracts/pcv/PCVUniswapDeposit.sol	ef62c37e17980298938c40d5d23c6b41e368e374c66bd32b9c1776689f6102b7
PCD	projects/contracts/pcv/PCVVenusDeposit.sol	b45d7e7ff787be13354e804f0e2fc952e54667fcf6aa352a17dc29a66b95798f

ID	File	SHA256 Checksum
RUM	projects/contracts/redemption/RedemptionUnit.sol	e0a8ac3a5dab03d9ae404e50ddf322dadda4c21894ab4ec5af04f76749bb77c4
XMR	projects/contracts/redemption/XMSRedemptionUnit.sol	adc5d37a11b62025eff96fb048480d5f1047a6ad50837ad9aa0dc91bb3bb0e43
CRM	projects/contracts/refs/CoreRef.sol	2ead42cd70c84114b4445244081ae57cf893c55d1441ec11ad4e3e7cae54e12c
ORM	projects/contracts/refs/OracleRef.sol	a9ec38bfe4de255255592291ce3b5ae6298a391b842a4e415f770430448c24b4
UAO	projects/contracts/refs/UniAndOracleRef.sol	459f8f0cf7e90700b2872bf5b48e28f6c77fa19a12ff75cb3636eff57d4903ff
URM	projects/contracts/refs/UniRef.sol	76357c5d3b6d59ff2f0e99a6d0ac7c8dcd89aafc87d73667d170a007f3096b04
MMM	projects/contracts/stake/MarsMaker.sol	413dcd1f7bcbbf8db9fac48140c38760de8880f0ababf07231029f59c9b5a16f
MSM	projects/contracts/stake/MarsStake.sol	4ded6ad8a53dd8bbd1a069864b5050f20a6a71e80f500f2c19b668eb9cea0745
MSE	projects/contracts/stake/MarsStakeReward.sol	75ed474512f854129d83508a7060d7a1df3c25f678dfe2758b70464e392e1c50
MSC	projects/contracts/swap/MarsSwapERC20.sol	13ed11feacba2dc04b8483a33c43c8bf70fe8555441c8d93971c4b85457c349a
MSF	projects/contracts/swap/MarsSwapFactory.sol	679f9037c19ef3299c97f9eca614df3e700ab113fd53817e953131a58b0f0fc7
MSK	projects/contracts/swap/MarsSwapPair.sol	495ceecee21349e585126083523efa078ce7b54f699b3077b62965a51e2b7beb
SMM	projects/contracts/swap/SwapMining.sol	8c4688d934f7fe790b6372a0d78c827be706b039e31656364040152db85cb600
DME	projects/contracts/utils/Delegatee.sol	198893ff3c22bc48471f7b635ef428b6b6bb0c24f8244a1dfb8c386fe93b3ef3
LTM	projects/contracts/utils/LinearTokenTimelock.sol	415a5ea0adcd22fdabcdb435861b9bde109d7d22d0511ad4ddd120dbbf128f0d
MME	projects/contracts/utils/Multicall.sol	e5358cba7021fc35225ef77a437b3609ebd4f923040738c8fa149dc78dbee328

ID	File	SHA256 Checksum
PTM	projects/contracts/utils/PeriodTokenTimelock.sol	9f41fa74aa1299346b464acd1f6810bf59aeb361a2412b47638772658d3dad74
STM	projects/contracts/utils/StraightTokenTimelock.sol	86fd88e34c2ae9435ed44ba0383615b9029126a9c0161717ede2125d07f75418
TMC	projects/contracts/utils/Timed.sol	a43237e18e12ddd61a033dd36cd8c642ed984dd8e66d762d78ad16e84398f902

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**2** (5.56%)
🟨 **Medium**	**1** (2.78%)
🟨 **Minor**	**25** (69.44%)
🟦 **Informational**	**8** (22.22%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Privileged Ownership on State Management	**Centralization / Privilege**	● **Medium**	Partially Resolved
GLOBAL-02	Privileged Ownership on Asset Management	**Centralization / Privilege**	● **Major**	Partially Resolved
GLOBAL-03	Third Party Dependencies	Control Flow	● Minor	Acknowledged
GLOBAL-04	Return value not handled	Volatile Code	● Informational	Acknowledged
BNV-01	Return value not handled	Volatile Code	● Minor	✓ Resolved
BUD-01	Unnecessary `payable` modifier	Volatile Code	● Minor	✓ Resolved
BUD-02	Uninitialized state variables	Volatile Code	● Minor	✓ Resolved
BUD-03	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
BUD-04	Variable Could be Declared as `constant`	Coding Style	● Informational	✓ Resolved
BUP-01	Unnecessary `payable` modifier	Volatile Code	● Minor	✓ Resolved
BUP-02	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
BUP-03	Return value is never assigned	Volatile Code	● Minor	✓ Resolved
BUS-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
CME-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved

ID	Title	Category	Severity	Status
DME-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
DME-02	Suicidal Delegatee Contract	Volatile Code	● Informational	✓ Resolved
IDO-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
IMO-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
IMO-02	Check of purchaseCap can be bypassed	Volatile Code	● Minor	ⓘ Acknowledged
LMM-01	Compare variable to boolean constant	Gas Optimization	● Informational	✓ Resolved
LMM-02	Recommended Explicit Pool Validity Checks	Logical Issue	● Informational	✓ Resolved
LTM-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
LTT-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
MMM-01	Missing zero address validation	Volatile Code	● Informational	✓ Resolved
MSM-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
MSR-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
MSR-02	Incompatibility With Deflationary Tokens	Logical Issue	● Minor	ⓘ Acknowledged
RUM-01	Unnecessary `payable` modifier	Volatile Code	● Minor	✓ Resolved
RUM-02	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved
SMM-01	Compare variable to boolean constant	Gas Optimization	● Informational	✓ Resolved
SMM-02	Large Trade Quantity	Logical Issue	● Minor	ⓘ Acknowledged
SMO-01	Dangerous Time-based Calculation	Volatile Code	● Minor	ⓘ Acknowledged
TME-01	MINIMUM_DELAY equal to 0	**Centralization / Privilege**	● **Major**	✓ Resolved
USD-01	Missing event emissions	Language Specific	● Informational	✓ Resolved

ID	Title	Category	Severity	Status
USD-02	Inconsistency `onlyBurner` permission	Control Flow	● Minor	✓ Resolved
XMR-01	Wrong EIP20 Application	Volatile Code	● Minor	✓ Resolved

GLOBAL-01 | Privileged Ownership on State Management

Category	Severity	Location	Status
Centralization / Privilege	● Medium	Global	Partially Resolved

Description

When deploying the contract, the Mars Ecosystem team will set the `Governor` address for the protocol. This protocol admin will be granted high-level permissions, becoming a `Governor`, `Minter`, `Burner`, `PCV_Controller` and `Guardian` in the system.

These roles have different abilities to update the state variables and manage the assets. This finding focuses on the management of the critical state variables.

In terms of updating the critical state variables, the `Governor` will be able to:

1. Add and revoke roles
2. Change Chainlink address
3. Set the fee
4. Set the allocate incentive amount
5. Set the allocation of incoming protocol controlled value(PCV)
6. Change oracle addresses
7. Set the maximum and minimum stable price of allocating PCV
8. Set the USDM and XMS token address
9. Set the genesis group address
10. Create swap mining pool and update the `allocPoint` of the pool
11. Pause and unpause functions

In terms of updating the critical state variables, the `Guardian` will be able to:

1. Pause and unpause functions

In terms of updating the critical state variables, the `PCV_Controller` will be able to:

1. Set the address of LP mining master
2. Leave PCV Deposit supply

Recommendation

We advise the client to handle these privileged roles carefully avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

Alleviation

[Mars Ecosystem Team]:

a) Timelock with 24 hours latency has been deployed.

b) Multi-sig is currently used with the team and advisors acting as co-signers.

c) DAO module will be applied once the governance tokens are sufficiently distributed to the public.

Here is the address of Timelock:

- 0xC35a8BdBB93abFAb362aF6dC3383cD2c6aEA6cBc

Here are the list of Multisig wallets:

- 0xe40b248a2a1c3d8f01b2324379a708cabbce0720
- 0xb0187445719656254f3f196f5aa9b72203556174
- 0x7da267ff4db18d4ba01a826b284cf34affd004b8
- 0x8d7205ee6c929529ecac3374bf9a4885381e988a
- 0xabfa05df381aa2e7a59b908d9bcb4fc266350469
- 0x066a763e737cea02dddd3d9fa186657a06bbdbf1
- 0xc13e199b32b5e758519b4d67d50e8bbf40f365a7

GLOBAL-02 | Privileged Ownership on Asset Management

Category	Severity	Location	Status
Centralization / Privilege	● Major	Global	Partially Resolved

Description

When deploying the contract, the Mars Ecosystem team will set the `Governor` address for the protocol. This protocol admin will be granted high-level permissions, becoming a `Governor`, `Minter`, `Burner`, `PCV_Controller` and `Guardian` in the system.

These roles have different abilities to update the state variables and manage the assets. This finding focuses on the management of assets in contracts.

In terms of managing assets in contracts, the `Governor` will be able to:

1. Send XMS tokens from treasury to an address
2. Send any token from treasury to an address
3. Transfer locked token to the beneficiary

In terms of managing assets in contracts, the `Minter` will be able to:

1. Mint XMS tokens
2. Mint USDM tokens

In terms of managing assets in contracts, the `Burner` will be able to:

1. Burn USDM tokens from a given address

In terms of managing assets in contracts, the `Guardian` will be able to:

1. Remove liquidity when the token price is within the input threshold
2. Withdraw LP mining for BUSD PCV

In terms of managing assets in contracts, the `PCV_Controller` will be able to:

1. Withdraw tokens from PCV allocations
2. Remove liquidity when the token price is within the input threshold
3. Harvest from PCV allocations
4. Deposit and withdraw LP minings with a given amount

Recommendation

We advise the client to handle these privileged roles carefully avoid any potential hack. We also advise the client to consider the following solutions:

1. `Timelock` with reasonable latency for community awareness on privileged operations;
2. Multisig with community-voted 3rd-party independent co-signers;
3. DAO or Governance module increasing transparency and community involvement.

Alleviation

`Burner` role and the privileged function `burnFrom()` are removed in commit hash `ee66e4c945f56690c6c787e99bd1b621d1ef7682`.

[Mars Ecosystem Team]:

a) Timelock with 24 hours latency has been deployed.

b) Multi-sig is currently used with the team and advisors acting as co-signers.

c) DAO module will be applied once the governance tokens are sufficiently distributed to the public.

Here is the address of Timelock:

- 0xC35a8BdBB93abFAb362aF6dC3383cD2c6aEA6cBc

Here are the list of Multisig wallets:

- 0xe40b248a2a1c3d8f01b2324379a708cabbce0720
- 0xb0187445719656254f3f196f5aa9b72203556174
- 0x7da267ff4db18d4ba01a826b284cf34affd004b8
- 0x8d7205ee6c929529ecac3374bf9a4885381e988a
- 0xabfa05df381aa2e7a59b908d9bcb4fc266350469
- 0x066a763e737cea02dddd3d9fa186657a06bbdbf1
- 0xc13e199b32b5e758519b4d67d50e8bbf40f365a7

GLOBAL-03 | Third Party Dependencies

Category	Severity	Location	Status
Control Flow	● Minor	Global	ⓘ Acknowledged

Description

The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen. In addition, upgrades of 3rd parties are possible to lead to severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logics of Mars require the interaction with Chainlink, Uniswap, WETH, VBNB, etc. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Mars Team]: We only use well-known and fully-tested 3rd party dependencies such as Chainlink etc. We are constantly monitoring the statuses of those 3rd parties. In addition, there are no plans of introducing new 3rd party dependencies.

GLOBAL-04 | Return value not handled

Category	Severity	Location	Status
Volatile Code	● Informational	Global	ⓘ Acknowledged

Description

The return value of a couple of functions in the project is not properly handled. Lines affected are listed below:

- BUSDUniswapPCVDeposit.sol: L74, L83, L97
- IDO.sol: L43, L68
- MarsSwapRouter.sol: L377

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle the return values properly if needed by the business logic. Otherwise, if the return values are intended to be ignored, recommend properly documenting or commenting this behavior.

According Solidity Documentation: The `require` function should be used to ensure valid conditions that cannot be detected until execution time. This includes conditions on inputs or return values from calls to external contracts.

Alleviation

[Mars Team]: The return values of the methods need to be analyzed in specific business scenarios, not all checks are required. Some return values are the result, which means those are just to elaborate that the functions are called, so the checks do not need to be imposed on the results. There are a series of checks already done in the process.

BNV-01 | Return value not handled

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/pcv/BNBVenusPCVDeposit.sol: 67	✓ Resolved

Description

The return value of a couple of functions in the project is not properly handled.

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

BUD-01 | Unnecessary `payable` modifier

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/genesis/BUSDGenesisGroup.sol: 75	✓ Resolved

Description

The "purchase" function is not expected to receive any BNB token in the function call. The `payable` modifier is unnecessary and might cause users to lose money if they send a transaction with `msg.value` larger than 0.

Recommendation

Remove the `payable` modifier

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

BUD-02 | Uninitialized state variables

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/genesis/BUSDGenesisGroup.sol: 22, 39	✓ Resolved

Description

Variable `cap` and `launchTimestamp` are used in the contract, but they don't have a value assigned.

Recommendation

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

Alleviation

Fixed in commit hash `ff5738777fc757396cc72838a4c0cbc74ff68f63`.

BUD-03 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/genesis/BUSDGenesisGroup.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

BUD-04 | Variable Could be Declared as `constant`

Category	Severity	Location	Status
Coding Style	● Informational	projects/contracts/genesis/BUSDGenesisGroup.sol: 22	✓ Resolved

Description

State variable `cap` is only assigned in initialization and never changed.

Recommendation

Recommend declaring `cap` is constant if by design it would never be re-assigned.

Alleviation

Fixed in commit hash `440f43185e12d98788e214e276551f348eb47dff`.

BUP-01 | Unnecessary `payable` modifier

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/pcv/BUSDUniswapPCVDeposit.sol: 40	✓ Resolved

Description

The "purchase" function is not expected to receive any BNB token in the function call. The `payable` modifier is unnecessary and might cause users to lose money if they send a transaction with `msg.value` larger than 0.

Recommendation

Remove the `payable` modifier

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

BUP-02 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/pcv/BUSDUniswapPCVDeposit.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

BUP-03 | Return value is never assigned

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/pcv/BUSDUniswapPCVDeposit.sol: 59	✓ Resolved

Description

Function `_getAmountUSDMToDeposit()` has a return value declared, `amountUSDM`. However, this value is never assigned, thus the return value of this function would always be empty.

Alleviation

Fixed in commit hash `440f43185e12d98788e214e276551f348eb47dff`.

BUS-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/bondingcurve/BUSDBondingLCurve.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

CME-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/core/Core.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

DME-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/utils/Delegatee.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

DME-02 | Suicidal Delegatee Contract

Category	Severity	Location	Status
Volatile Code	● Informational	projects/contracts/utils/Delegatee.sol	✓ Resolved

Description

Contract `Delegatee` does not have functions to handle Ethers it receive. Function `withdraw()` would transfer all `xms` to the owner address and then the call of `selfdestruct` would send all Ether to the owner address as well. After the self-destruction, the new transfers to the old `Delegatee` address would be locked.

Recommendation

According to Solidity Documentation: If you want to deactivate your contracts, you should instead disable them by changing some internal state which causes all functions to revert. This makes it impossible to use the contract, as it returns Ether immediately.

Alleviation

Fixed in commit hash `8f72334a5a64e74a92de362560ab5639c4df4107`.

IDO-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/genesis/IDO.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

IMO-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/genesis/IMO.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

IMO-02 | Check of purchaseCap can be bypassed

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/genesis/IMO.sol: 85~89	ⓘ Acknowledged

Description

In function `purchase()`, the check of `purchaseCap` is checking the balance of an address using the `balanceOf()` function of `ERC20`. However, if a user first purchase some tokens, and then transfer the tokens to a new address, the balance of the user is less than the `purchaseCap`, which means the user can purchase again and again.

Recommendation

Recommend storing the value of purchased tokens of each address in the contract, instead of using the external ERC20's `balanceOf()` call.

Alleviation

[Mars Team]: It would be difficult to avoid users having multiple accounts. This one would be ignored.

LMM-01 | Compare variable to boolean constant

Category	Severity	Location	Status
Gas Optimization	● Informational	projects/contracts/liquidity/LiquidityMiningMaster.sol: 59	✓ Resolved

Description

`poolExistence[_lpToken]` is compared to boolean constants at multiple locations. Boolean constants can be used directly and do not need to be compare to true or false.

Recommendation

We recommend removing the equality to the boolean constant.

Alleviation

Fixed in commit hash `ff5738777fc757396cc72838a4c0cbc74ff68f63`.

LMM-02 | Recommended Explicit Pool Validity Checks

Category	Severity	Location	Status
Logical Issue	● Informational	projects/contracts/liquidity/LiquidityMiningMaster.sol: 211, 106, 242, 180, 151, 273	✓ Resolved

Description

There's no sanity check to validate if a pool is existing.

Recommendation

We advise the client to adopt following modifier `validatePoolByPid` to functions `setPool()`, `deposit()`, `withdraw()`, `emergencyWithdraw()`, `pendingXMS()` and `updatePool()`.

```
modifier validatePoolByPid(uint256 _pid) {
    require (_pid < poolInfo . length , "Pool does not exist") ;
    _;
}
```

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

LTM-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/utils/LinearTokenTimelock.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

LTT-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/dao/LinearTokenTimelockDelegator.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

MMM-01 | Missing zero address validation

Category	Severity	Location	Status
Volatile Code	● Informational	projects/contracts/stake/MarsMaker.sol: 47~48	✓ Resolved

Description

Lacks of zero address check on the `_bar` and `_weth` variables in the contract constructor.

Recommendation

Check that the `_bar` and `_weth` address is not zero.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

MSM-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/stake/MarsStake.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

MSR-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/liquidity/MarsSwapRouter.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

MSR-02 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Logical Issue	● Minor	projects/contracts/liquidity/MarsSwapRouter.sol: 112, 113, 145, 403, 430	ⓘ Acknowledged

Description

The users add, remove or swap LP tokens into the router, and the `mint`, `burn` and `swap` operations are performed. When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. As a result, the amount inconsistency will occur and the transaction may fail due to the validation checks.

Recommendation

We advise the client to regulate the set of LP tokens supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[Mars Team]:

a) Deflationary tokens are supported.

b) It is allowed for users to add tokens and provide liquidity.

c) For deflationary tokens, if the transactions failed, users can manually adjust the slippages to broadcast the transactions.

RUM-01 | Unnecessary `payable` modifier

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/redemption/RedemptionUnit.sol: 41	✓ Resolved

Description

The "purchase" function is not expected to receive any BNB token in the function call. The `payable` modifier is unnecessary and might cause users to lose money if they send a transaction with `msg.value` larger than 0.

Recommendation

Remove the `payable` modifier

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

RUM-02 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/redemption/RedemptionUnit.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

SMM-01 | Compare variable to boolean constant

Category	Severity	Location	Status
Gas Optimization	● Informational	projects/contracts/swap/SwapMining.sol: 58	✓ Resolved

Description

`poolExistence[_lpToken]` is compared to boolean constants at multiple locations. Boolean constants can be used directly and do not need to be compare to true or false.

Recommendation

We recommend removing the equality to the boolean constant.

Alleviation

Fixed in commit hash `ff5738777fc757396cc72838a4c0cbc74ff68f63`.

SMM-02 | Large Trade Quantity

Category	Severity	Location	Status
Logical Issue	● Minor	projects/contracts/swap/SwapMining.sol: 292	ⓘ Acknowledged

Description

The value of `quantity` can be manipulated through a transaction with flash loan. External users only needs to pay trade fee and flash loan fee to generate a large value of `quantity`.

Recommendation

Recommend properly documenting this design decision and making sure the community understand this behavior is allowed. Also consider directly pointing out that it is acceptable by design for the contract to accept massive transactions that come with the flash loan.

Alleviation

[Mars Team]:

a) Calling through flash loan is allowed. There are no difference between the call of flash loan and normal transactions, they are both regular and valid transactions

b) Increasing the share of mining by increasing the volume of transactions is a valid business attribute by design

SMO-01 | Dangerous Time-based Calculation

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/oracle/SwapMiningOracle.sol: 121	ⓘ Acknowledged

Description

In function `consult()`,`price[0/1]Average` would be updated based on the variable `timeElapsed`. It is possible to call update() first and then immediately call consult(). In this scenario, the value of `timeElapsed` would be rather small, and thus `priceAverage` could get a relatively large value.

For the TWAP value, a larger `timeElapsed` value could provide a more stable `priceAverage`.

Recommendation

Short term: If by design it is accepted to having an extra small `timeElapsed`, e.g. `1`, please properly documenting this design decision and making sure the community understand it is allowed.

Long term: Recommend leveraging more testing or formal verification to calculate or prove a relatively small TWAP is valid and correct.

Alleviation

[Mars Team]:

a) This mechanism has already been tested by other well-known protocols.

b) Trade mining pools are not randomly added. It is comparatively more costly to manipulate prices.

c) Tokens traded out must be in the whitelist of trade mining.

d) Mining is a continuous process and cannot be quickly arbitrated through flash loans.

TME-01 | MINIMUM_DELAY equal to 0

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	projects/contracts/dao/Timelock.sol: 42	✓ Resolved

Description

The MINIMUM_DELAY variable indicates the minimal time delay before a queued transaction can be executed. The MINIMUM_DELAY equals zero defeats the purpose of using a timelock contract.

Recommendation

Consider increasing the MINIMUM_DELAY to at least 24 hours.

Alleviation

Fixed in commit hash `ff5738777fc757396cc72838a4c0cbc74ff68f63`.

USD-01 | Missing event emissions

Category	Severity	Location	Status
Language Specific	● Informational	projects/contracts/base/USDMToken.sol: 48, 60	✓ Resolved

Description

Several actions such as token minting and burning don't emit events.

Recommendation

Consider emitting events for all state changing functions, including token mining and burning.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

USD-02 | Inconsistency `onlyBurner` permission

Category	Severity	Location	Status
Control Flow	● Minor	projects/contracts/base/USDMToken.sol: 60	✓ Resolved

Description

Function `burnFrom()` is restricted by modifier `onlyBurner`, but function `burn()` does not have this modifier. Please make sure the permission of `onlyBurner` is consistent.

Alleviation

Fixed in commit hash `ee66e4c945f56690c6c787e99bd1b621d1ef7682`.

XMR-01 | Wrong EIP20 Application

Category	Severity	Location	Status
Volatile Code	● Minor	projects/contracts/redemption/XMSRedemptionUnit.sol	✓ Resolved

Description

According to EIP-20, functions `transfer()`, `transferFrom()`, and `approve()` should always have a `bool` return value, for the ERC20 caller to handle, as the callers must not assume that `false` is never returned.

Alleviation

Fixed in commit hash `384574397dfef6ba7262a36ba9766a01d58c6096`.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

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

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

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS

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

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

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

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

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

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

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

About

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

CERTIK