Making Web3 Space Safer for Everyone

KALOS

Baby Shark Universe Token (BSU Token)

Security Assessment

Published on : 4 Aug. 2023

Version v1.0

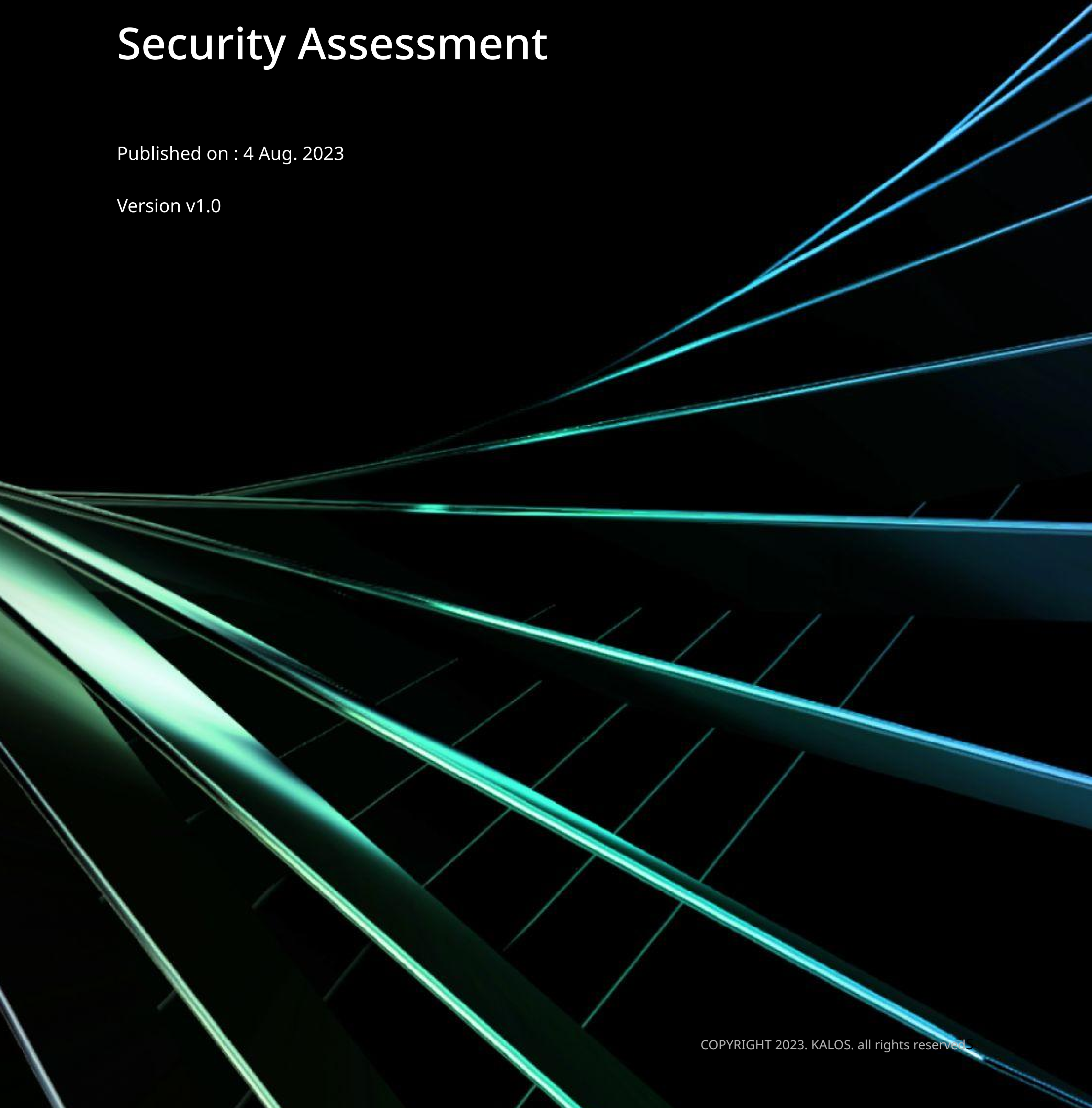

TABLE OF CONTENTS

Executive Summary

Purpose of this report

This report has been prepared by the KALOS team to audit the security of the BSU token smart contract developed by the Baby Shark Universe. The audit focused on verifying the implementation and design of the smart contract as stated in the publicly available materials provided by the Baby Shark Universe, with an emphasis on security. In order to ensure the proper functioning of BSU token, the following aspects were tested:

- Project availability issues like Denial of Service.
- Suitable access control
- Suitable implementation of ERC20 specifications.

Codebase Submitted for the Audit

The codes used in this Audit can be found on GitHub (https://github.com/Babysharkuniverse/BSU_contracts).

The last commit of the code used for this Audit is "41da8e45424d75e1c4c9229e0ce566af5a276881".

This project is a simple ERC-20 token which has access control with the OWNER_ROLE and is pausable. The person who has OWNER_ROLE is an admin role.

Scope

└── BSU_Token.sol

Audit Timeline

Date	Event
2023/08/02	Audit Initiation (BSU Token)
2023/08/04	Delivery of v1.0 report.

Inherent Risks Identified

The following inherent risks have been identified for the BSU token project. The relevant details are described in the Inherent Risks Analysis section.

Risk Type	Status
Access Controls	● the owner can add owner of this smart contract ● the owner can remove owner of this smart contract ● the owner can pause and unpause this smart contract ● the owner can burn the tokens ● the owner can distribute tokens of its own.
Token Supply	● The initial supply is 1,500,000,000 tokens with 18 decimals ● Total supply can be changed when the owner burns the token.
Confidential Transaction	● The contract code under security audit does not have any functionality to hide transactions.

Security Assessment Findings

KALOS found 0 Critical, 0 High, 0 medium and 0 Low severity issues. There are 0 Tips issues explained that would improve the code’s usability or efficiency upon modification.

Methodology

The KALOS team conducted an audit of the project scope using a combination of manual and automated security testing. Manual testing is the process of finding flaws in process implementation and logic, while automated testing helps to improve code coverage of the project and quickly identify items that do not follow best practices and standards. During the audit, we used the following steps and associated tools.

Structure and purpose study → Manual code review and execution → Logic/connection/function graphing (solgraph) →

Manual testing with test scripts → Static Analysis (Slither, MythX, Pyrometer, etc.) →

Test environment deployment testing (Hardhat, Foundry, etc.)

These steps cover various aspects of the security audit process, including reviewing and assessing the project's Smart Contract source code, security vulnerabilities, test coverage, protocol spec verification, access control analysis, and other risks. Based on the findings, we documented the results of the security audit and presented recommendations and conclusions.

Smart Contract Overview

Documentation Overview

Baby Shark Universe is a metaverse platform based on the Polygon ecosystem. Using a user-friendly gaming environment, it aims to change how it interacts with the virtual world. Baby Shark Universe aspires to become a safe, decentralized, and transparent blockchain platform based on blockchain technology and the immersive user interface of 2D and 3D games. The BSU token can be acquired by Mining via GameFi and selling custom maps. Also, this token aims to play custom maps within the world, crafting and editing, and registering custom maps within the world - entry to the land.

Functional Description

• **ERC-20**
The audited contracts in this project is a simple ERC-20 token contract which supports Pausable and owner role.

• **Pausable**
The audited token contract allows it to be paused/unpaused by the owner.

• **AccessControlEnumerable**
The audited contracts in this project is using AccessControlEnumerable for addOwner function and removeOwner function.

Inherent Risks Analysis

Access Controls

Token Owner

- BabysharkUniverseToken.distributeTokens(): The owners can distribute its token.
- BabysharkUniverseToken.pause(): Pause token transfer.
- BabysharkUniverseToken.unpause(): Unpause token transfer.
- BabysharkUniverseToken.burn(): The owners can burn their own tokens.
- BabysharkUniverseToken.addOwner(): The owner can grant admin role(OWNER_ROLE).
- BabysharkUniverseToken.removeOwner(): The owner can revoke admin role(OWNER_ROLE).

Token Supply Analysis

The initial token supply of the BSU token is 1,500,000,000 tokens with 18 decimals. The owners of this contract are the only people who can burn their own tokens. And, there is no private or public mint function. This means, The total supply cannot be increased.

Confidential Transaction

The contract code under security audit does not have any functionality to hide transactions.

Dependency

The contract under security audit has dependencies on the following libraries.

BSU_Token.sol

```
@openzeppelin/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/access/AccessControlEnumerable.sol
@openzeppelin/contracts/security/Pausable.sol
```

Security Audit Findings

No issue found.

DISCLAIMER

This report does not guarantee investment advice, the suitability of the business models, and codes that are secure without bugs. This report shall only be used to discuss known technical issues. Other than the issues described in this report, undiscovered issues may exist such as defects on the main network. In order to write secure codes, correction of discovered problems and sufficient testing thereof are required.

Appendix

Test Results - Static Analysis

```
INFO:Detectors:
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)
        - inverse = (3 * denominator) ^ 2
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#116)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#120)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#121)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#122)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#123)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#124)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - denominator = denominator / twos
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#101)
        - inverse *= 2 - denominator * inverse
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#125)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
performs a multiplication on the result of a division:
        - prod0 = prod0 / twos (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#104)
        - result = prod0 * inverse (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
AccessControlEnumerable._grantRole(bytes32,address)
(lib/openzeppelin-contracts/contracts/access/AccessControlEnumerable.sol#52-55) ignores return value by
_roleMembers[role].add(account)
(lib/openzeppelin-contracts/contracts/access/AccessControlEnumerable.sol#54)
AccessControlEnumerable._revokeRole(bytes32,address)
```

```
(lib/openzeppelin-contracts/contracts/access/AccessControlEnumerable.sol#60-63) ignores return value by
_roleMembers[role].remove(account)
(lib/openzeppelin-contracts/contracts/access/AccessControlEnumerable.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
INFO:Detectors:
Strings.toString(uint256) (lib/openzeppelin-contracts/contracts/utils/Strings.sol#19-39) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/Strings.sol#25-27)
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/Strings.sol#31-33)
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#62-66)
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#85-92)
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#99-108)
EnumerableSet.values(EnumerableSet.Bytes32Set)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#219-229) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#224-226)
EnumerableSet.values(EnumerableSet.AddressSet)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#293-303) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#298-300)
EnumerableSet.values(EnumerableSet.UintSet)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#367-377) uses assembly
        - INLINE ASM (lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#372-374)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity are used:
        - Version used: ['^0.8.0', '^0.8.9']
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/access/AccessControl.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/access/AccessControlEnumerable.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/access/IAccessControl.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/access/IAccessControlEnumerable.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/security/Pausable.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/utils/Context.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/utils/Strings.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/utils/introspection/ERC165.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/utils/introspection/IERC165.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/utils/math/SignedMath.sol#4)
        - ^0.8.0 (lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#5)
        - ^0.8.9 (src/BSU.sol#3)
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Context._msgData() (lib/openzeppelin-contracts/contracts/utils/Context.sol#21-23) is never used and
should be removed
EnumerableSet._values(EnumerableSet.Set)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#153-155) is never used and should
be removed
EnumerableSet.add(EnumerableSet.Bytes32Set,bytes32)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#169-171) is never used and should
be removed
EnumerableSet.add(EnumerableSet.UintSet,uint256)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#317-319) is never used and should
be removed
EnumerableSet.at(EnumerableSet.Bytes32Set,uint256)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#207-209) is never used and should
be removed
EnumerableSet.at(EnumerableSet.UintSet,uint256)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#355-357) is never used and should
be removed
EnumerableSet.contains(EnumerableSet.AddressSet,address)
```

```
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#260-262) is never used and should
be removed
EnumerableSet.contains(EnumerableSet.Bytes32Set,bytes32)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#186-188) is never used and should
be removed
EnumerableSet.contains(EnumerableSet.UintSet,uint256)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#334-336) is never used and should
be removed
EnumerableSet.length(EnumerableSet.Bytes32Set)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#193-195) is never used and should
be removed
EnumerableSet.length(EnumerableSet.UintSet)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#341-343) is never used and should
be removed
EnumerableSet.remove(EnumerableSet.Bytes32Set,bytes32)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#179-181) is never used and should
be removed
EnumerableSet.remove(EnumerableSet.UintSet,uint256)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#327-329) is never used and should
be removed
EnumerableSet.values(EnumerableSet.AddressSet)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#293-303) is never used and should
be removed
EnumerableSet.values(EnumerableSet.Bytes32Set)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#219-229) is never used and should
be removed
EnumerableSet.values(EnumerableSet.UintSet)
(lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#367-377) is never used and should
be removed
Math.average(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#34-37) is never
used and should be removed
Math.ceilDiv(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#45-48) is never
used and should be removed
Math.log10(uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#252-284) is never used
and should be removed
Math.log10(uint256,Math.Rounding) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#290-295) is
never used and should be removed
Math.log2(uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#199-235) is never used and
should be removed
Math.log2(uint256,Math.Rounding) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#241-246) is
never used and should be removed
Math.log256(uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#303-327) is never used
and should be removed
Math.log256(uint256,Math.Rounding) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#333-338)
is never used and should be removed
Math.max(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#19-21) is never
used and should be removed
Math.min(uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#26-28) is never
used and should be removed
Math.mulDiv(uint256,uint256,uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#55-134)
is never used and should be removed
Math.mulDiv(uint256,uint256,uint256,Math.Rounding)
(lib/openzeppelin-contracts/contracts/utils/math/Math.sol#139-145) is never used and should be removed
Math.sqrt(uint256) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#152-183) is never used and
should be removed
Math.sqrt(uint256,Math.Rounding) (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#188-193) is
never used and should be removed
SignedMath.abs(int256) (lib/openzeppelin-contracts/contracts/utils/math/SignedMath.sol#37-42) is never
used and should be removed
SignedMath.average(int256,int256)
(lib/openzeppelin-contracts/contracts/utils/math/SignedMath.sol#28-32) is never used and should be
removed
SignedMath.max(int256,int256) (lib/openzeppelin-contracts/contracts/utils/math/SignedMath.sol#13-15) is
```

```
never used and should be removed
SignedMath.min(int256,int256) (lib/openzeppelin-contracts/contracts/utils/math/SignedMath.sol#20-22) is
never used and should be removed
Strings.equal(string,string) (lib/openzeppelin-contracts/contracts/utils/Strings.sol#82-84) is never
used and should be removed
Strings.toHexString(uint256) (lib/openzeppelin-contracts/contracts/utils/Strings.sol#51-55) is never
used and should be removed
Strings.toString(int256) (lib/openzeppelin-contracts/contracts/utils/Strings.sol#44-46) is never used
and should be removed
Strings.toString(uint256) (lib/openzeppelin-contracts/contracts/utils/Strings.sol#19-39) is never used
and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/access/AccessControl.sol#4) allows old
versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/access/AccessControlEnumerable.sol#4) allows
old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/access/IAccessControl.sol#4) allows old
versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/access/IAccessControlEnumerable.sol#4)
allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/security/Pausable.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/ERC20.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/IERC20.sol#4) allows old
versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/token/ERC20/extensions/IERC20Metadata.sol#4)
allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/Context.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/Strings.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/introspection/ERC165.sol#4) allows old
versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/introspection/IERC165.sol#4) allows
old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/math/Math.sol#4) allows old versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/math/SignedMath.sol#4) allows old
versions
Pragma version^0.8.0 (lib/openzeppelin-contracts/contracts/utils/structs/EnumerableSet.sol#5) allows
old versions
Pragma version^0.8.9 (src/BSU.sol#3) allows old versions
solc-0.8.19 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Parameter BabysharkUniverseToken.burn(uint256)._amount (src/BSU.sol#49) is not in mixedCase
Reference:
https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-convention
s
INFO:Detectors:
BabysharkUniverseToken.constructor() (src/BSU.sol#14-20) uses literals with too many digits:
        - _mint(msg.sender,1500000000 * 10 ** uint256(decimals())) (src/BSU.sol#19)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits
INFO:Slither:src/BSU.sol analyzed (16 contracts with 88 detectors), 73 result(s) found
```

Severity Level

CRITICAL	Must be addressed as a vulnerability that has the potential to seize or freeze substantial sums of money.
HIGH	Has to be fixed since it has the potential to deny users compensation or momentarily freeze assets.
MEDIUM	Vulnerabilities that could halt services, such as DoS and Out-of-Gas, need to be addressed.
LOW	Issues that do not comply with standards or return incorrect values
TIPS	Tips that makes the code more usable or efficient when modified

Difficulty Level

	Low	**Medium**	**High**
Privilege	anyone	Miner/Block Proposer	Admin/Owner
Capital needed	Small or none	Gas fee or volatile as price change	More than exploited amount
Probability	100%	Depend on environment	Hard as mining difficulty

Vulnerability Category (Smart Contract)

Arithmetic	• Integer under/overflow vulnerability • floating point and rounding accuracy
Access & Privilege Control	• Manager functions for emergency handle • Crucial function and data access • Count of calling important task, contract state change, intentional task delay
Denial of Service	• Unexpected revert handling • Gas limit excess due to unpredictable implementation
Miner Manipulation	• Dependency on the block number or timestamp. • Frontrunning
Reentrancy	•Proper use of Check-Effect-Interact pattern. •Prevention of state change after external call • Error handling and logging.
Low-level Call	• Code injection using delegatecall • Inappropriate use of assembly code
Off-standard	• Deviate from standards that can be an obstacle of interoperability.
Input Validation	• Lack of validation on inputs.
Logic Error/Bug	• Unintended execution leads to error.
Documentation	•Coherency between the documented spec and implementation
Visibility	• Variable and function visibility setting
Incorrect Interface	• Contract interface is properly implemented on code.

End of Document