CERTIK

Security Assessment

ASM 3

Jun 28th, 2022

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- CON-01 : Centralization Related Risks
- CON-02 : Uninitialized State Variable
- COV-01 : Divide Before Multiply
- TES-02 : Missing Error Messages
- UTI-01 : Ineffective `isContract()` Check

Optimizations

- CON-03 : Improper Usage of `public` and `external` Type
- TES-01 : Variables That Could Be Declared as `constant`

Appendix

Disclaimer

About

Summary

This report has been prepared for ASM 3 to discover issues and vulnerabilities in the source code of the ASM 3 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 Formal Verification 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	ASM 3
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/altered-state-machine/genome-mining-contracts-audit
Commit	10c283633bc408095640f50892e1cb6ebac5d06b

Audit Summary

Delivery Date	Jun 28, 2022 UTC
Audit Methodology	Static Analysis, Formal Verification

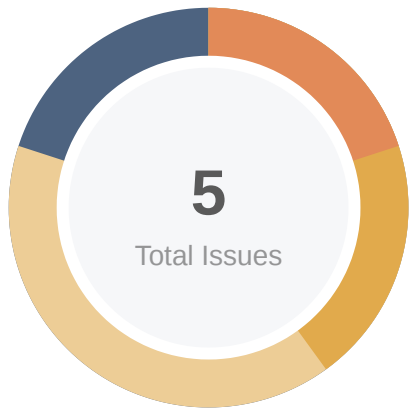
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	1	0	0	1	0	0	0
Medium	1	0	0	1	0	0	0
Minor	2	0	0	0	0	0	2
Optimization	2	0	0	1	0	0	1
Informational	1	0	0	1	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ICB	contracts/helpers/IConverter.sol	798dc4f33de6dc7c0c70c0cf84eb212d18180d19d32c4b7d3d30f6214ab955ed
ISB	contracts/helpers/IStaking.sol	149ed2427c7806687d9ce6aa1fa81da5dde7dfbd09ec6493680f04e9604ae572
PCB	contracts/helpers/PermissionControl.sol	440f7eb4a1b67e64feced9144f742fd5a4b921c5680abd3bacc5e8c1b66d31b6
TCB	contracts/helpers/TimeConstants.sol	57ed73c89b071c053c70aa3b310cb5fe6ddbb19aee2802eb2e45c89cb0bbb9b0
UTI	contracts/helpers/Util.sol	5fb18da6c995ba48cd3526f007b05accc52cdd7fe48bb73b4a38b5fe7e170349
SSB	contracts/StakingStorage.sol	dd670f20224720a10dba878da5d94fa9abb948474a691c5557adeef3467d353b
STA	contracts/Staking.sol	5444f27aa9dd9298ab55cb16ad6c249b0cd09ceec87eae3c6d0e07764900ce3c
ESB	contracts/EnergyStorage.sol	8bfcd45dac178f6f73badd790de7eedac193c7bbbce52a239ded22a4cae14dc8
CON	contracts/Controller.sol	3d55262dfeef5c676ab0c452cb0da3419a7ed2172484a899680251639b70952d
COV	contracts/Converter.sol	9955542478e9eadbc720a5424704b72b4f7c29d5fdf658e723294acf8d2e935a

Findings



Critical	**0** (0.00%)
Major	**1** (20.00%)
Medium	**1** (20.00%)
Minor	**2** (40.00%)
Informational	**1** (20.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CON-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
CON-02	Uninitialized State Variable	Coding Style	Minor	Resolved
COV-01	Divide Before Multiply	Mathematical Operations	Minor	Resolved
TES-02	Missing Error Messages	Coding Style	Informational	Acknowledged
UTI-01	Ineffective `isContract()` Check	Volatile Code	Medium	Acknowledged

CON-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	contracts/Controller.sol: 46, 186, 199, 205, 209, 213, 217, 221, 225, 229, 234; contracts/Converter.sol: 126, 190, 198, 220, 243, 282, 290, 298, 306; contracts/EnergyStorage.sol: 32, 60, 68, 76; contracts/Staking.sol: 61, 108, 112, 116; contracts/StakingStorage.sol: 45, 99, 106	Acknowledged

Description

In the contract `Controller.sol`, the role `MANAGER_ROLE` has authority over the following functions:

- **init**
- **upgradeContracts**
- **setManager**
- **setController**
- **setStakingLogic**
- **setAstoStorage**
- **setLpStorage**
- **setConverterLogic**
- **setEnergyStorage**
- **pause**
- **unpause**

Any compromise to the `MANAGER_ROLE` account may allow a hacker to take advantage of this authority and basically control everything such as by calling admin functions and managing and upgrading contracts addresses.

In the contract `Converter.sol`, the role `MANAGER_ROLE` has authority over the following functions:

- **setUser**
- **addPeriods**
- **addPeriod**
- **updatePeriod**

Any compromise to the `MANAGER_ROLE` account may allow a hacker to take advantage of this authority and set or update the desired user in the administration, add and update periods.

In the contract `Staking.sol`, the role `MANAGER_ROLE` has authority over the **withdraw** function and any compromise to the `MANAGER_ROLE` account may allow a hacker to take advantage of this authority and withdraw the available tokens from the wallet.

In the contract `Converter.sol`, the role `USER_ROLE` has authority over the **useEnergy** function and any compromise to the `USER_ROLE` account may allow a hacker to take advantage of this authority and consume the desired amount of energy from the wallet.

In the contract `Converter.sol`, the role `CONTROLLER_ROLE` has authority over the **init**, **setManager**, **setController**, **pause** and **unpause** functions and any compromise to the `CONTROLLER_ROLE` account may allow a hacker to take advantage of this authority and use the admin functions and set the desired manager and controller.

In the contract `EnergyStorage.sol`, the role `CONTROLLER_ROLE` has authority over the **init**, **setController**, **pause** and **unpause** functions and any compromise to the `CONTROLLER_ROLE` account may allow a hacker to take advantage of this authority and use the admin functions and set the desired controller, including pausing and unpausing the contract.

In the contract `Staking.sol`, the role `CONTROLLER_ROLE` has authority over the **init**, **setManager**, **setController**, **pause** and **unpause** functions and any compromise to the `CONTROLLER_ROLE` account may allow a hacker to take advantage of this authority and use the admin functions and set the desired controller, including pausing and unpausing the contract.

In the contract `StakingStorage.sol`, the role `CONTROLLER_ROLE` has authority over the **init** and **setController** functions and any compromise to the `CONTROLLER_ROLE` account may allow a hacker to take advantage of this authority and use init function to update role and logic of stakers, unpause the contract, and set the address of new controller.

In the contract `EnergyStorage.sol`, the role `CONVERTER_ROLE` has authority over the **increaseConsumedAmount** function and any compromise to the `CONVERTER_ROLE` account may allow a hacker to take advantage of this authority and increase the desired consumed energy for address `addr`.

In the contract `StakingStorage.sol`, the role `STAKER_ROLE` has authority over the **updateHistory** function and any compromise to the `STAKER_ROLE` account may allow a hacker to take advantage of this authority and update the history by saving stakes into the storage.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

`[ASM]`: We started to use DAO and Multisig contracts to manage the mentioned functions. The multisig is controlled by ASM team and requires at least 3 signers to execute the transaction. The DAO is controlled by the community and requires a successful vote to execute.

CON-02 | Uninitialized State Variable

Category	Severity	Location	Status
Coding Style	Minor	contracts/Controller.sol: 19, 33, 112; contracts/Staking.sol: 127, 150	Resolved

Description

One or more state variables are used without being initialized in the constructor.

File: contracts/Controller.sol (Line 19, Contract `Controller`)

```
    Controller public controller_;
```

- `controller_` is never initialized, but used in `Controller._setController`.

File: contracts/Staking.sol (Line 33, Contract `Staking`)

```
    mapping(uint256 => string) private _tokenName;
```

- `_tokenName` is never initialized, but used in:
  - `Staking.stake`
  - `Staking.unstake`

Recommendation

We recommend initializing the state variables at declaration or in the constructor. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

Alleviation

`[CertiK]`: The team heeded the advice and revolved the finding in the commit d825ad9740b6c05d6adb6d9b5ad9201441fbfa1c

COV-01 | Divide Before Multiply

Category	Severity	Location	Status
Mathematical Operations	Minor	contracts/Converter.sol: 97~98	Resolved

Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

Recommendation

Consider ordering multiplication before division to avoid loss of precision.

Alleviation

`[CertiK]`: The team heeded the advice and revolved the finding in the commit 49c05b73b681a3f1f325cc050f897ef7f27621ce

TES-02 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	tests/Staking.test.sol: 388; tests/StakingStorage.test.sol: 156	Acknowledged

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

`[ASM]`: It's unit test helper functions which we will not fix it in current version

UTI-01 | Ineffective `isContract()` Check

Category	Severity	Location	Status
Volatile Code	Medium	contracts/helpers/Util.sol: 43	Acknowledged

Description

The implementation of the `isContract` check can not cover all scenarios. The check can be bypassed if the call is from the constructor of a smart contract or when the contract is destroyed. Because, in that case, the codesize will also be zero.

The "isContract" function in the OpenZeppelin "Address" library uses the same implementation, but comments mention that "it's unsafe to rely on the check and it can be bypassed". Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Address.sol

Recommendation

It is recommended to add the additional `msg.sender == tx.origin` check to cover all the scenarios. Do note that the check still works for the current EVM (London) version, but future updates to the EVM or EIP (ex. EIP-3074) might cause the check to become ineffective.

```
modifier notContract() {
    require((!_isContract(msg.sender)) && (msg.sender == tx.origin), "contract not
allowed");
    _;
}

function _isContract(address addr) internal view returns (bool) {
    uint256 size;
    assembly {
        size := extcodesize(addr)
    }
    return size > 0;
}
```

Alleviation

`[ASM]`: We only use the function to check if our own contracts are valid. It's not used to validate the msg.sender in our case.

Optimizations

ID	Title	Category	Severity	Status
CON-03	Improper Usage Of `public` And `external` Type	Gas Optimization	Optimization	Resolved
TES-01	Variables That Could Be Declared As `constant`	Gas Optimization	Optimization	Acknowledged

CON-03 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Optimization	contracts/StakingStorage.sol: 45; contracts/mocks/MockedERC20.sol : 17, 21	Resolved

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

Alleviation

`[CertiK]`: The team heeded the advice and revolved the finding in the commit d825ad9740b6c05d6adb6d9b5ad9201441fbfa1c

TES-01 | Variables That Could Be Declared As `constant`

Category	Severity	Location	Status
Gas Optimization	Optimization	tests/Staking.test.sol: 39, 40, 41, 42, 44, 45, 46; tests/StakingStorage.test.sol: 39, 40, 41, 43, 44, 45	Acknowledged

Description

The linked variables could be declared as `constant` since these state variables are never modified.

Recommendation

We recommend to declare these variables as `constant`.

Alleviation

`[ASM]`: It's unit test helper functions which we will not fix it in current version

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Volatile Code

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

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.