DApp Developers and Smart Contract Auditors

SMART CONTRACT SECURITY AUDIT

of

GPU.NET CONTRACTS

Smart Contract Audit of GPU.net

May 21st, 2024 | v. 1.1

TABLE OF CONTENT

AUDIT INTRODUCTION

Auditing Firm	SecureDApp Auditors
Audit Architecture	SecureDApp Auditing Standard
Language	Solidity
Client Firm	GPU.net
Website	https://gpu.net/
Report Date	May 21st, 2024

About GPU.net

Democratize access to high-performance computing resources, making them readily available for diverse applications ranging from data analysis and scientific research to AI development and beyond.

AUDIT DOCUMENT

Name	Smart Contract Code Review and Security Analysis Report for GPU.net
Approved By	Himanshu Gautam| CTO at SecureDApp
Type	GPU.net
Platform	EVM
Language	Solidity
Changelog	21.5.2024 – Final Review

AUDIT SCOPE

The scope of this report is to audit the smart contract source code of GPU.net.
Our client provided us with one smart contract.

- gpu.sol

The contract was written in Solidity and based on the OpenZeppelin library.
The contract allows providers to register themselves and rent out their machines onchain. The users borrow machines for a particular time period by paying in G_Points (native chain currency)..

After initial research, we agreed to perform the following tests and analyses as part of our well-rounded audit:

- Smart contract behavioral consistency analysis
- Test coverage analysis
- Penetration testing: checking against our database of vulnerabilities and simulating manual attacks against the contracts
- Static analysis
- Manual code review and evaluation of code quality
- Analysis of GAS usage
- Contract analysis with regards to the host network

Final Review Scope

Repository	https://github.com/brahmGAN/gpu-contracts/tree/main/Contracts/GAN-Contracts
Commit	dc55fd9e287fc825adc9bfafc91d3d7d96f2668e
Functional Requirements	Partial documentation provided. README.md
Technical Requirements	Partial documentation provided. README.md
Contracts Addresses	Not Deployed yet

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to asset loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions.
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to asset loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution.
Informational	Issue listed to improve understanding, readability and quality of code

All statuses which are identified in the audit report are categorized here for the reader to review:

Status Type	Definition
Open	Risks are open.
Acknowledged	Risks are acknowledged, but not fixed.
Resolved	Risks are acknowledged and fixed.

AUDIT SUMMARY

The SecureDApp team has performed a line-by-line manual analysis and automated review of smart contracts. Smart contracts were analyzed mainly for common contract vulnerabilities, exploits, and manipulation hacks. According to the audit:

Status	Critical	High	Medium	Low	Informative- Gas Optimisation
Open	0	0	0	0	0
Acknowledged	3	1	0	1	3
Resolved	0	1	0	1	0

AUDIT METHODOLOGY

SecureDApp scans contracts and reviews codes for common vulnerabilities, exploits, hacks and back- doors. Mentioned are the steps used by SecureDApp to audit smart contracts:

a. Smart contract source code reviewal:
    i. Review of the specifications, sources, and instructions provided to SecureDApp to make sure we understand the audit scope, intended business behavior, overall architecture, and project's goal.
    ii. Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.
b. Test coverage analysis: (Unit testing)
    i. Test coverage analysis is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
c. Static analysis:
    i. Run a suite of vulnerability detectors to find security concerns in smart contracts with different impact levels.
d. Symbolically executed tests: (SMTChecker testing) (Taint analysis)
    i. Symbolic execution is analyzing a program to determine what inputs cause each part of a program to execute.
    ii. Check for security vulnerabilities using static and dynamic analysis
e. Property based analysis (Fuzz tests)(Invariant testing)
    i. Run the execution flow multiple times by generating random sequences of calls to the contract.
    ii. Asserts that all the invariants hold true for all scenarios.
f. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
g. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Automated 5S frameworks used to assess the smart contract vulnerabilities

- Consensys Tools
- SWC Registry
- Solidity Coverage
- Open Zeppelin Code Analyzer
- Solidity Shield Scan

We have audited the smart contracts for commonly known and more specific vulnerabilities. Below is the list of smart contract tests, vulnerabilities, exploits, and hacks:

ID	Description	Status
EEA 3.3	Oracle Manipulation	N/A
EEA 3.3	Bad Randomness - VRF	N/A
S60	Assembly Usage	Passed
S59	Dangerous usage of block.timestamp	Passed
EEA 3.7	Front-Running Attacks	N/A
EEA 3.7	Back-Running Attacks	N/A
EEA 3.7	Sandwich Attacks	N/A
DASP	Gas Griefing Attacks	Passed
DASP	Force Feeding	Passed
SCSVS V2	Access Control	-
DASP	Short Address Attack	Passed
DASP	Checks Effects Interactions	Passed
EEA 4.1	No Self-destruct	Passed
SCSVS V14	Decentralized Finance Checks	Passed

Slither Tests	Checks for ERC's conformance	Passed
Coverage	Unit tests with 100% coverage	-
Gas Reporter	Gas usage & limitations	-
Echidna Tests	Malicious input handling	Passed
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	Passed
SWC-103	Floating Pragma	-
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELF-DESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed

SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-134	Message call with the hardcoded gas amount	Passed
SWC-135	Code With No Effects (Irrelevant/Dead Code)	**Informational**
SWC-136/SCSVS V3	Unencrypted Private Data On-Chain	Passed

SYSTEM OVERVIEW

The GPU Network is a decentralized graphics processing unit on demand infrastructure that powers the next generation of Generative AI, Web3 Metaverses, High-end graphics rendering and cryptocurrency mining.

GPU.Net aims to build a decentralized platform that connects GPU providers to GPU consumers. GPU providers can attach their machine to the GPU.Net platform and earn GPU tokens for their idle time or total compute contribution. GPU consumers can access the vast infrastructure of GPU.Net by paying in tokens while ensuring scalability, affordability and security of their data and models.

Privileged roles

1. Owner_Role:
    a. Manage ownership of contract functionalities and upgrade features.

Risk

1. The impact of the owner role being compromised would have a huge impact on the protocol.
2. Centralization risk is the most common cause of cryptography asset loss
3. Compromising the Owner Role may lead to all system asset loss.

FINDINGS

Centralization Risk

Centralization risk is the most common cause of dapp's hacks. When a smart contract has an active contract ownership, the risk related to centralization is elevated. There are some well-intended reasons to be an active contract owner, such as:

- Contract owners can be granted the power to pause() or lock() the contract in case of an external attack.
- Contract owners can use functions like, include(), and exclude() to add or remove wallets from fees, swap checks, and transaction limits. This is useful to run a presale, and to list on an exchange.

Authorizing a full centralized power to a single body can be dangerous. Unfortunately, centralization related risks are higher than common smart contract vulnerabilities. Centralization of ownership creates a risk of rug pull scams, where owners cash out tokens in such quantities that they become valueless. Most important question to ask here is, how to mitigate centralization risk? Here's SecureDApp's recommendation to lower the risks related to centralization hacks:

- Smart contract owner's private key must be carefully secured to avoid any potential hack.
- Smart contract ownership should be shared by multi-signature (multi-sig) wallets.
- Smart contract ownership can be locked in a contract, user voting, or community DAO can be introduced to unlock the ownership.

GPU.NET's Centralization Status

- GPU smart contract has Contract Ownership Role assigned to a single wallet.

STATIC ANALYSIS REPORT

```
| Symbol | Meaning |
|:--------:|-----------|
| 🛑 | Function can modify state |
| 💵 | Function is payable |
| Contract | Type | Bases | | |
|:----------:|:-------------------:|:----------------:|:----------------:|:---------------:|
| └ | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
||||||
| **IGPU** | Interface | |||
||||||
| **IERC721** | Interface | |||
| └ | balanceOf | External ❗ | |NO❗ |
||||||
| **GPU** | Implementation | IGPU, OwnableUpgradeable, UUPSUpgradeable |||
| └ | _authorizeUpgrade | Internal 🔒 | 🛑 | onlyOwner |
| └ | initialize | Public ❗ | 🛑 | initializer |
| └ | withdraw | Public ❗ | 🛑 | onlyOwner |
| └ | addGpuType | Public ❗ | 🛑 | onlyOwner |
```

```
| └ | updateGpuPrice | Public ❗️ | 🛑  | onlyOwner |
| └ | addQueen | Public ❗️ | 💵 | haveNft isStakedAddress |
| └ | addProvider | External ❗️ | 💵 | haveNft isStakedAddress |
| └ | updateProviderStatus | External ❗️ | 🛑  |NO❗️ |
| └ | providerDrillRequest | External ❗️ | 🛑  |NO❗️ |
| └ | addConsumer | External ❗️ | 🛑  |NO❗️ |
| └ | createJob | Public ❗️ | 💵 |NO❗️ |
| └ | drillTest | Internal 🔒 |   | |
| └ | calculateCost | Internal 🔒 |   | |
| └ | updateAssignedJob | Public ❗️ | 🛑  |NO❗️ |
| └ | reassignQueen | External ❗️ | 🛑  |NO❗️ |
| └ | updateAssignQueen | Internal 🔒 |   | |
| └ | healthCheckBundle | External ❗️ | 🛑  |NO❗️ |
| └ | healthCheckReport | Internal 🔒 | 🛑  | |
| └ | healthCheckTest | Internal 🔒 |   | |
| └ | setTickSeconds | Public ❗️ | 🛑  | onlyOwner |
| └ | setMachineId | Public ❗️ | 🛑  | onlyOwner |
| └ | setNftAddress | Public ❗️ | 🛑  | onlyOwner |
| └ | setStakeAmount | Public ❗️ | 🛑  | onlyOwner |
```

```
| └ | setMinDrillTestValue | Public ❗️ | 🛑  | onlyOwner |

| └ | setMinProviderAvailability | Public ❗️ | 🛑  | onlyOwner |

| └ | setMaxProviderUnavailability | Public ❗️ | 🛑  | onlyOwner |

| └ | setLatencyPeriod | Public ❗️ | 🛑  | onlyOwner |

| └ | setUserID | Public ❗️ | 🛑  | onlyOwner |

| └ | checkQueenLastCheck | Public ❗️ |   |NO❗️ |

| └ | getProviderStatus | Public ❗️ |   |NO❗️ |

| └ | getConsumerJobs | Public ❗️ |   |NO❗️ |

| └ | getProviders | Public ❗️ |   |NO❗️ |

| └ | getGpuTypes | Public ❗️ |   |NO❗️ |

| └ | getGpuPrice | Public ❗️ |   |NO❗️ |

| └ | getAssignedProviders | Public ❗️ |   |NO❗️ |

| └ | getDrillProvider | Public ❗️ |   |NO❗️ |

||||||
```

MANUAL REVIEW

Identifier	Definition	Severity
CEN-01	Centralization privileges of Marketplace Owner	Critical

Centralized privileges are listed below:

- Owner Role:
  - Manage Withdraw Funds, Add/Update GPU Types/Price
  - Set NFT Address, Staking Amount and other system parameters: tick seconds, min drill test value etc.
  - Contract Upgrades

RECOMMENDATION

Usage of OpenZeppelin Access Control framework instead of Ownership to allow Multiowner access based on different roles and avoid single point of failure. Use MultiSig wallets services for important authentications like contract upgrade access.

Access control privileges must be authenticated and their private keys should be secured carefully. Usage of Multi-Sig wallet for authorisation is recommended. Please refer to CENTRALIZED PRIVILEGES for a detailed understanding.

Status: Acknowledged

Identifier	Definition	Severity
CEN-02	Use of proxy and upgradeable contracts	Critical

Contract upgradeability allows privileged roles to change current contract implementation.

RECOMMENDATION

Test and validate the current contract thoroughly before deployment. Future contract upgradeability negatively elevates centralization risk.

Status: Acknowledged

Identifier	Definition	Severity
CEN-03	Missing Unit and End to End test cases for Smart Contract	Critical

Contracts lack unit and end-to-end test cases, leaving critical functionalities unvalidated and risking incomplete use case coverage. This gap increases the likelihood of undiscovered bugs and compromises the contracts' integrity.

RECOMMENDATION

Implementing comprehensive test suites and adhering to testing best practices to mitigate these risks effectively. Try to add unit test cases covering all edge patterns and achieve solidity coverage of 98%.

Status: Acknowledged

Identifier	Definition	Severity
HGH-01	Incorrect Interface Import	High

Change Interface import from NEWIGPU.sol to IGPU.sol.

RECOMMENDATION

Fixed the interface import

Status: Resolved

Identifier	Definition	Severity
HGH-02	Fluidity in system roles	High

The system allows for the interchangeability between the roles of a Queen and a Provider. This fluidity in roles introduces a potential vulnerability wherein a Queen, entrusted with strategic decision-making, can also become a Provider, and vice versa. Such role interchangeability might lead to a compromise in the system's integrity, as it blurs the delineation of duties and authority, potentially resulting in conflicts of interest or mismanagement of resources.

RECOMMENDATION

Implement stakeholder KYC protocols to prevent authority misuse and promptly forfeit staked amounts upon misconduct detection.

Status: Acknowledged

Identifier	Definition	Severity
LOW-01	Missing events	Low

- Contract is missing events for important updates like:
    - setStakeAmount
    - setNftAddress

RECOMMENDATION

There should be an event emitted to record the important changes made to contract parameters.

Status: Resolved

Identifier	Definition	Severity
LOW-02	Version Pragma Recommendation	Low

- Avoid floating pragma statements.
- Solidity versions >=0.8.20 (gpu.sol) may be too recent to be fully trusted.

RECOMMENDATION

Consider deploying with fixed version ^0.8.18 to ensure compatibility and reliability

Status: Acknowledged

Identifier	Definition	Severity
INF-01	Contract Size Limitation Issue	Informational

The contract exceeds the permissible limit of 24576 bytes on the Ethereum Chain, potentially rendering it undeployable on the mainnet.

RECOMMENDATION

Shorten Require error messages to reduce contract size.

Utilize the updateAssignQueen() function at lines 161, 205, and 241 to avoid duplication and streamline code efficiency.

Redundant Code: In line 239, the required statement checking queensList.length > 0 is redundant, as line 242 will revert if the condition is true.

Status: Acknowledged

Identifier	Definition	Severity
INF-02	Unclear Payment Currency Info	Informational

The platform currency, referred to as GPoints, is actually implementing logic for the native currency of the deployed chain in the contract at lines 114, 143, and 235, causing confusion regarding the currency denomination.

RECOMMENDATION

Clarify the currency denomination within the contract documentation to avoid confusion among users and ensure clear understanding of the platform's currency system.

Status: Acknowledged

Identifier	Definition	Severity
INF-03	Gas Optimisations	Informational

RECOMMENDATIONS

- Combine Storage for Structs:
    - Optimize storage by combining structs to fit within 32 byte slots.
- Enhance efficiency by initializing the NFT contract instance only once, reducing redundant contract instantiation.
- Improve storage efficiency by creating a struct gpuInfo to avoid duplicate storage of gpuType string in gpuTypes and gpuPrices.
- Streamline logic by replacing isStaked with a check on stakes[address] > 0, reducing gas consumption.
- Enhance efficiency by incorporating queenStakings within the Queen struct to avoid separate mappings.
- Increase gas efficiency by switching from public to external visibility where internal calls are not required.
- Onchain Storage Optimization: Store GPU details on IPFS and save the IPFS link to the contract. This reduces on-chain storage costs and enhances contract performance.

Status: Acknowledged

DISCLAIMER

SecureDApp Auditors provides the easy-to-understand audit of solidity source codes (commonly known as smart contracts).

The smart contract for this particular audit was analyzed for common contract vulnerabilities, and centralization exploits. This audit report makes no statements or warranties on the security of the code. This audit report does not provide any warranty or guarantee regarding the absolute bug-free nature of the smart contract analyzed, nor do they provide any indication of the client's business, business model or legal compliance. This audit report does not extend to the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. Cryptographic tokens are emergent technologies, they carry high levels of technical risks and uncertainty. 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. This audit report could include false positives, false negatives, and other unpredictable results.

CONFIDENTIALITY

This report is subject to the terms and conditions (including without limitations, description of services, confidentiality, disclaimer and limitation of liability) outlined in the scope of the audit provided to the client. This report should not be transmitted, disclosed, referred to, or relied upon by any individual for any purpose without SecureDApp's prior written consent.

NO FINANCIAL ADVICE

This audit report does not indicate the endorsement of any particular project or team, nor guarantees its security. No third party should rely on the reports in any way, including to make any decisions to buy or sell a product, service or any other asset. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. This audit report should not be used in any way to make decisions around investment or involvement. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. for avoidance of doubt, services, including any associated audit reports or materials, shall not be considered or relied upon as any form of financial, tax, legal, regulatory, or other advice.

TECHNICAL DISCLAIMER

All services, audit reports, smart contract audits, 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, SecureDApp hereby disclaims all warranties, whether expressed, implied, statutory, or otherwise with respect to services, audit report, or other materials. Without limiting the foregoing, SecureDApp specifically disclaims all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement, and all warranties arising from the course of dealing, usage, or trade practice. Without limiting the foregoing, SecureDApp makes no warranty of any kind that all services, audit reports, smart contract audits, or other materials, or any products or results of the use thereof, will meet the client's or any other individual'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.

TIMELINESS OF CONTENT

The content contained in this audit report is subject to change without any prior notice. SecureDApp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following the publication.

LINKS TO OTHER WEBSITES

This audit report provides, through hypertext or other computer links, access to websites and social accounts operated by individuals other than SecureDApp. Such hyperlinks are provided for your reference and convenience only and are the exclusive responsibility of such websites' and social accounts' owners. You agree that SecureDApp is not responsible for the content or operation of such websites and social accounts and that SecureDApp shall have no liability to you or any other person or entity for the use of third-party websites and social accounts. You are solely responsible for determining the extent to which you may use any content at any other websites and social accounts to which you link from the report.

ABOUT SECUREDAPP

SecureDApp Auditor provides intelligent blockchain solutions. SecureDapp is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. SecureDapp's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy to use.

SecureDApp is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 6+ core team members, and 10+ supporting contributors. SecureDApp provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.

To learn more, visit : https://securedapp.in/

To view our audit portfolio, visit : github.securedapp.in

To book an audit, message : securedapp.telegram

Securedapp.in

Securedapp_Linkedin

Securedapp_Telegram