Smart Contract Security Audit

For MFPS project

Author: Crypto CTF OÜ, Awalcon Security Team

Date: 2023. April 11.

Table of Contents

Summary of the Audit

Awalcon, the web3 security team of Cryto CTF OÜ, conducted a security smart contract audit on the MFPS project's code base (*https://metaplayers.gg/*), which included the new token, old token migration, stake, two NFT smart contracts, related interfaces and examples.

After assessing the smart contracts' security from the provided source code and specifications, Awalcon identified high, low and informational level risks. These risks were reported to the MFPS team during the audit, and the necessary fixes were implemented.

Overall, we can conclude that the project was well-developed, and even though vulnerabilities were discovered, except two, they were limited in scope. All the high risk vulnerabilities have been fixed before the final deployment of the live project. The updated code base has been rechecked by Awalcon team to verify and confirm the fixes.

Audit overview

Start date of the audit: 2023.04.01.

Report date: 2023.04.11. (extended on request)

Project website: https://metaplayers.gg/
Platform: Solidity / Binance Smart Chain
Code author: Silur
Audited project packages:

> ***sha256sum mfps.tar # Original code base received for audit***
4bfd1baed8927d9b1e77ecacf6aa5e47b53cbf5a9becca1b4a50012794c0e515

>***sha256sum mfps-fix1.tar # The package received after Silur's fixes***
cecd8b7b4c48953476280c7dd9811ea9be1c7d45ae06afb2122b7cf5df2d8c2d

> ***sha256sum mfps_latest.tar # Changed after first round, and fixed swap***
99e67b8d285f82367ee13fd55e1fcb2225dec2c0f8da8475d1450fc8e922f71c

> ***Note:*** *the swap error was fixed in this version, contract got redeployed.*

Smart contracts in scope:

- FPSMigrate.sol
- MFPS.sol
- Migrations.sol
- NetworkBuidlerNFT.sol
- PatronProgram.sol
- Staking.sol
- VipNft.sol
- VIPStaking.sol

Overall result: **mfps_update.tar - Passed**

Auditors:

six ~ *six@cryptoctf.org / PGP 450F 4AC8 0BD8*
G ~ g@cryptoctf.org

Objective and methodology

The objective of the security assessment is to gain insight into the security of the smart contracts listed in the scope.

Code review main check items:

- Line-by-line audit
- Business logic
- Data consistency
- Coding style violations
- Gas usage
- Reentrancy
- Test with automated tools:
  - Static analysis with Slither
  - Fuzzing with Echidna
  - Symbolic execution with Manticore and/or Mythril

Further documents incorporated in the methodology:

- Smart Contract Weakness Classification Registry - https://swcregistry.io/
- https://github.com/miguelmota/solidity-audit-checklist

Symbolic execution test cases

List of test scenarios

- Caller can redirect execution to arbitrary bytecode locations
- Caller can write to arbitrary storage locations
- Control flow depends on a predictable environment variable
- Control flow depends on tx.origin
- Any sender can withdraw ETH from the contract account
- Assertion violation
- External call to another contract
- Integer overflow or underflow
- Multiple external calls in the same transaction
- State change after an external call
- Contract can be accidentally killed by anyone
- Return value of an external call is not checked
- A user-defined assertion has been triggered

Risk Classifications

Critical: Vulnerabilities that can lead to a loss of funds, impairment, or external control over the system or its function. We recommend that findings of this classification are fixed immediately.

High: Findings of this classification can impact the flow of logic and can cause direct disruption in the system and the project's organization. We recommend that issues of this classification are fixed as soon as possible.

Medium: Vulnerabilities of this class have impact on the flow of logic, but does not cause any disturbance that would halt the system or organizational continuity. We recommend that findings of this class are fixed nonetheless.

Low: Bugs, or vulnerability that have minimal impact and do not pose a significant threat to the project or its users. We recommend that issues of this class are fixed nonetheless because they increase the attack surface when your project is targeted by malicious actors.

Informational: Findings of this class have a negligible risk factor but refer to best practices in syntax, style or general security.

Audit results

Critical severity

No critical severity vulnerabilities have been found.

High severity

[FIXED] MFPS.sol - max_supply can increase above project promises

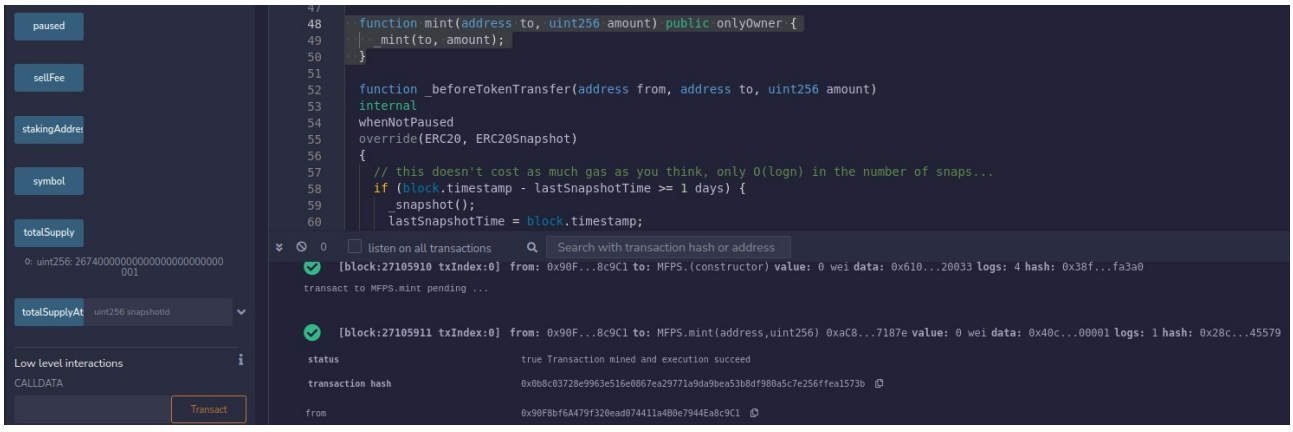
Description

The maximum supply of the MFPS tokens can be increased unintentionally by the onlyOwner. Because of some hardcoded values this can also break the calculations throughout the whole project (eg. DPR/APR).

Affected functions: mint() and mintStakingRewards()

Exploitation

Remix IDE can be used to trigger the exploit:



Proposed solution

Add require checks to the mint function so the max_supply cannot be changed. The mint is used in multiple parts of the system, they need to be fixed in all-together.

Reference

This vulnerability is specific to the MFPS.sol smart contract.

Medium severity

No medium severity vulnerabilities have been found.

Low severity

[ACCEPTED] Use of onlyOwner decreasing decentralization

Description:

It was found that the project have single point of failures in the system: the smart contract's onlyOwner modifier - single key controlling functionalities.

Impact:

In case the onlyOwner admin account is breached, the project might be taken down as a whole. It can happen through multiple scenarios, examples are the following:

- Stealing the devices physically that stores the private keys
- Exploitation of the system
- By human errors, losing the devices
- System errors, eg. ssd/disk failure and lack of usable backup
- Insider threat
- Incident of the owners of the devices and having no possibility to restore the private keys

Recommendations:

Implement decentralization for the admin functions.

References:

https://github.com/Qrucial/Voronoi

Threshold ECDSA: https://eprint.iacr.org/2019/114.pdf

[ACCEPTED] Visibility of functions could be stricter

Description:

If a function does not require to be called internally, it is possible to save gas costs and slightly improve security by changing their visibility from public to external.

Impact:

Function calls will cost less gas (both deploy and call times) and security will be slightly improved.

Recommendations:

Replace "public" to "external" in the functions listed:

FPSMigrate.sol: function enableMigration() public

MFPS.sol: function snapshot() public

MFPS.sol: function pause() public

MFPS.sol: function unpause() public

MFPS.sol: function setWhitelist(address addr, bool status) public

MFPS.sol: function setStakingAddress(address addr) public

MFPS.sol: function setFees(uint256 buy, uint256 sell) public

Migrations.sol: function setCompleted(uint completed) public

NetworkBuidlerNFT.sol: function safeMint(address to) public

PatronProgram.sol: function startVIP() public

PatronProgram.sol: function deposit(uint256 amount) public

PatronProgram.sol: function withdraw() public

Staking.sol: function startStaking() public

Staking.sol: function deposit(uint256 amount) public

Staking.sol: function unstake() public

Staking.sol: function withdraw(bool force) public

VipNft.sol: function setPatronContract(address addr) public

VIPStaking.sol: function startVIP() public

VIPStaking.sol: function deposit(uint256 amount) public

VIPStaking.sol: function withdraw() public

VIPStaking.sol: function transferReserves(address addr) public

References:

https://ezcook.de/2018/01/29/Gas-Used-by-Public-and-External-Function-in-Solidity/

https://ethereum.stackexchange.com/questions/19380/external-vs-public-best-practices

Informational

[ACCEPTED] Lack of comments regarding functionality

Description

The smart contracts have comments at some critical points, but not about the functions or general code logic.

It is also recommended to add NatSpec to the code.

Proposed solution

Follow the Solidity Coding style guide.

https://docs.soliditylang.org/en/latest/style-guide.html

[ACCEPTED] Lack of comments regarding functionality

Description

The smart contracts have comments at some critical points, but not about the functions or general code logic.

It is also recommended to add NatSpec to the code.

Proposed solution

Follow the Solidity Coding style guide.

https://docs.soliditylang.org/en/latest/style-guide.html

[ACCEPTED] Multiple versions of compiler used

Description

The project has multiple versions of compilers set.

Version used: ['>=0.5.0', '>=0.6.2', '^0.8.0', '^0.8.8', '^0.8.9']

Impact

Compiling the system have minor compilation inconsistencies.

Proposed solution

We propose either to accept this information.

Disclaimer

The list of findings and recommendations are summarized in the Audit Results.

The matters raised in this report are only those identified during the review and are not necessarily a comprehensive statement of all weaknesses that exist or all actions that might be taken. This work was performed under limitations of time and scope that are not potentially relevant to the actions of a malicious attack.

The review is based at a specific point in time, in an environment where both the systems and the threat profiles are dynamically evolving. It is therefore possible that vulnerabilities exist or will arise that were not identified during the review and there may or will have been events, developments and changes in circumstances subsequent to its issue.

The security analysis is purely based on the provided smart contracts alone. No other products or systems have been reviewed. The purpose of the audit is to identify issues related to the logic and quality of the code.

Regarding the MFPS migration process, the teams were working closely together as Awalcon was monitoring the process and changes were done in the code, also a new audit request was received. Though we have identified most bugs and the developer team fixed those, due to a last minute change a new bug was introduced by the developers. The new code was quick-tested against security vulnerabilities in less than a few hours, but due to stress on short deadline both sides missed a feature test scenario so a new deployment was needed (hence the project was deployed 2 times). The bug was not security, but feature related, however didn't allow the use of swap feature in a meaningful manner. - There is never full coverage for testing, but there is full dedication in bringing a high quality project, that is why we kept sitting on the project for days, barely sleeping during the migration. It is better we found the issue and fixed through an early redeployment, than leaving such a bug live and causing greater damages.

Contact

Awalcon Team - six

Website: *https://awalcon.org/*

E-mail: *six@cryptoctf.org*

Git: *https://git.hsbp.org/six*

PGP*: B1F7 B1D6 8838 98B4 2212 1D90 CA71 D1E4 078E 99C5*

Awalcon Team - G

Website: *https://awalcon.org/*

E-mail: *gabo@cryptoctf.org*

Crypto CTF OÜ – Audit request

Website: *https://cryptoctf.org/*

E-mail: *contact@cryptoctf.org*