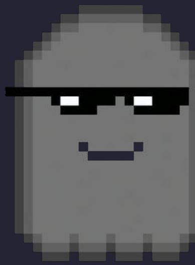
GUARDIAN AUDITS

SMART CONTRACT SECURITY AUDIT OF



Ultimate Fantoms

Summary

Audit Firm: Guardian Audits

Client Firm: Fantoms on Opera

Final Report Date May 23, 2022

Audit Summary

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- Ultimate Fantom's smart contracts have a LOW RISK SEVERITY
- Ultimate Fantom's smart contracts have an **ACTIVE OWNERSHIP**
- Important owner privileges – `setRoyaltyAddress`, `updateSpiritRouter`, `updatePaintRouter`, `setMintSize`, `sweepEthToAddress`

  Ultimate Fantom's smart contract owner has multiple "write" privileges. Centralization risk correlated to the active ownership is **MEDIUM**

Notice that the examined smart contracts are not resistant to internal exploit. For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

📜 Ultimate Fantom's contract address: **0x287986A4cdfC7957e9fc273e353995BC2A2E93aE**

📜 Royalty Splitter's contract address: **0x0A5298D3ff18359d946c7BC6A1e1DF8C86aD0A96**

🔗 Blockchain network: **Fantom Opera**

✅ Verify the authenticity of this report on Guardian's GitHub: https://github.com/guardianaudits

Table of Contents

Project Overview

Project Summary

Project Name	Ultimate Fantoms
Language	Solidity
Codebase	https://ftmscan.com/address/0x287986a4cdfc7957e9fc273e353995bc2a2e93ae#code
Commit	N/A

Audit Summary

Delivery Date	May 23, 2022
Audit Methodology	Static Analysis, Manual Review, Full Test Suite, Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	5	0	0	0	0	0
● Low	13	0	0	0	0	0

Audit Scope & Methodology

Scope

ID	File	SHA-1 Checksum
UF	UltimateFantoms.sol	D0E20190FD19CE048B5683CA246A1C8C33919AF0
RS	RoyaltySplitter.sol	60DB419906EAB591D7FA55650219B95FAAD44BC2

Methodology

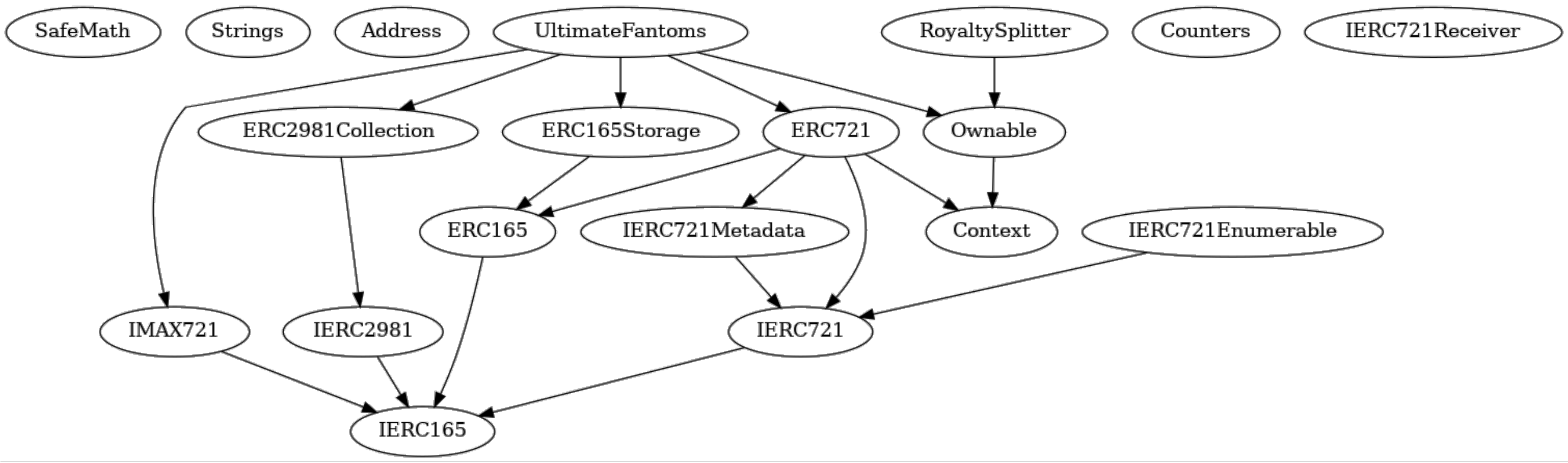
The auditing process pays special attention to the following considerations:

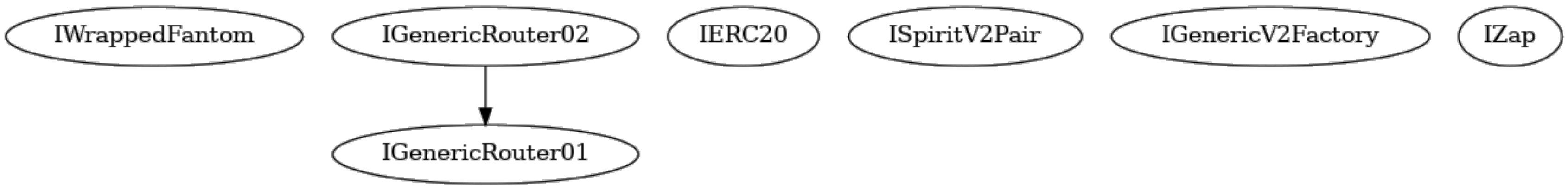
- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and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

Vulnerability Classifications

Vulnerability Level	Classification
🔴 Critical	Easily exploitable by anyone, causing loss/manipulation of assets or data.
🟠 High	Arduously exploitable by a subset of addresses, causing loss/manipulation of assets or data.
🟡 Medium	Inherent risk of future exploits that may or may not impact the smart contract execution.
🟢 Low	Minor deviation from best practices.

Inheritance Graph





Findings & Resolutions

ID	Title	Category	Severity	Status
UF-1	Centralization Risk	Centralization / Privilege	● Medium	Acknowledged
UF-2	DoS With Failed Call	DoS	● Medium	Acknowledged
UF-3	Random Manipulation	Tx Manipulation	● Medium	Acknowledged
UF-4	Mint Failure	Logical Error	● Medium	Resolved
UF-5	Price Inconsistency	Logical Error	● Medium	Resolved
UF-6	Using .transfer	Best Practices	● Low	Resolved
UF-7	Inaccurate Comments	Code Cleanliness	● Low	Resolved
UF-8	Unnecessary Code	Code Cleanliness	● Low	Resolved
UF-9	Repetitive Function Calls	Optimization	● Low	Resolved
UF-10	Mint Fee Manipulation	Fee Manipulation	● Low	Acknowledged
UF-11	Arbitrary Max Supply	Tokenomics	● Low	Acknowledged
UF-12	Unequal Minting Rewards	Logical Error	● Low	Resolved
UF-13	Mutability Modifiers	Mutability	● Low	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
UF-14	Function Visibility Modifiers	Optimization	● Low	Resolved
RS-1	Unnecessary Code	Code Cleanliness	● Low	Resolved
RS-2	Unequal Royalty Rewards	Logical Error	● Low	Resolved
RS-3	Repetitive Function Calls	Optimization	● Low	Resolved
RS-4	Mutability Modifiers	Mutability	● Low	Resolved

UF-1 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Medium	UltimateFantoms.sol	Acknowledged

Description

The `owner` address, `0x3e522051a9b1958aa1e828ac24afba4a551df37d`, is not a multi-sig and has potentially dangerous permissions for `renounceOwnership`, `transferOwnership`, `setRoyaltyAddress`, `setSpiritRouter`, `updatePaintRouter`, `setBaseURI`, `setMintSize`, `sweepEthToAddress`.

Recommendation

Make the `owner` a multi-sig and/or introduce a timelock for improved community oversight.

Resolution

Ultimate Fantoms: Acknowledged, contract ownership will be changed to the multisig at

`0x87f385d152944689f92Ed523e9e5E9Bd58Ea62ef`.

UF-2 | Denial-of-Service With Failed Call

Category	Severity	Location	Status
DoS	● Medium	UltimateFantoms.sol	Acknowledged

Description

`publicMint` relies on multiple external calls which can fail accidentally or deliberately. If just one consistently fails, users will not be able to mint.

Recommendation

Isolate external calls to another transaction(s). `wFTM` allocations could be distributed with a pull-over-push pattern.

Resolution

Ultimate Fantoms: Acknowledged, failed transactions can be resubmitted + the chance of

a failed call is low.

UF-3 | Random Manipulation

Category	Severity	Location	Status
Tx Manipulation	● Medium	UltimateFantoms.sol	Acknowledged

Description

The random function relies on weak sources of pseudo-randomness from only on-chain attributes. A validator node can manipulate the block.timestamp and therefore the random number. Therefore, the _sendTo address can be manipulated in favor of the validator.

Recommendation

Utilize the Randomness pattern to obtain on-chain randomness and avoid validator manipulation or obtain random numbers off-chain through an oracle.

Resolution

Ultimate Fantoms: Acknowledged in source code.

UF-4 | Mint Failure

Category	Severity	Location	Status
Logical Error	● Medium	UltimateFantoms.sol:1719	Resolved

Description

In publicMint, when performing _earnTo = random() % (_tokenIdCounter.current() +1), there is a possibility _earnTo is equivalent to _tokenIdCounter.current() which yields a tokenID for a token that does not exist yet. Therefore, the subsequent call to ownerOf will fail and the mint will revert.

Recommendation

Perform random() % _tokenIdCounter.current().

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

UF-5 | Price Inconsistency

Category	Severity	Location	Status
Logical Error	● Medium	UltimateFantoms.sol: 1594	Resolved

Description

In the `getPrice` function the stepwise price jumps do not account for the following `tokenIds`: 101, 301, 601, 1001, 1501, and 2301.

This is because each `if` statement utilizes `>` instead of `>=` when referring to these tokenIds.Therefore a mint for one of these `tokenIds` will mistakenly go to the else branch and charge 6 `FTM.`

Recommendation

Use `>=` or decrement the lower boundaries by one.

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

UF-6 | Using .transfer

Category	Severity	Location	Status
Best Practices	● Low	UltimateFantoms.sol:1829	Resolved

Description

`transfer()` comes with a fixed amount of gas.

Recommendation

Utilize `call()` with a success check.

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

UF-7 | Inaccurate Comments

Category	Severity	Location	Status
Code Cleanliness	● Low	UltimateFantoms.sol: 1712, 1672	Resolved

Description

On line 1712: `// random number between 0 to 4` is inaccurate as `_toEarn` takes on a random value of 0 or 1.

Additionally, on line 1672: `// 10%` is inaccurate as `_rndmAlloc` is calculated to be 15%.

Recommendation

Refactor comments to accurately reflect the code.

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

UF-8 | Unnecessary Code

Category	Severity	Location	Status
Code Cleanliness	● Low	UltimateFantoms.sol	Resolved

Description

Several functions such as setMintFees, enableMinting, disableMinting, setTeamMinting, minterTeamMintsRemaining, and minterTeamMintsCount serve no purpose.

In addition, variables such as enableMinter, _earnAmount, _mintFees, _teamMintSize, RNDM_TOKEN, bePATH1, bePATH2, BEETS_TOKEN, bPath1, bPath2, BRUSH_TOKEN, SPIRITSWAP_TOKEN, wCYBERs, wFTMOPRs, OPR, _beetsAlloc, _treasuryAlloc, _dfyAlloc, and _teamMintCounter go unused.

Recommendation

Remove unused code.

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

UF-9 | Repetitive Function Calls

Category	Severity	Location	Status
Optimization	● Low	UltimateFantoms.sol	Resolved

Description

In publicMint the random function is called several times, even though the random value is constant during each tx.

Recommendation

Compute the random value once.

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

UF-10 | Mint Fee Manipulation

Category	Severity	Location	Status
Fee Manipulation	● Low	UltimateFantoms.sol: 1667	Acknowledged

Description

Because the getPrice function is stepwise, anyone can mint 10 tokens as if they were all in a lower cost bracket while potentially only 1 was.

Recommendation

Compute the mint fee for each token, account for mints that traverse the fee increase, or accept the manipulation.

Resolution

Ultimate Fantoms: Acknowledged, protocol loss will be minimal if exploited.

UF-11 | Arbitrary Max Supply

Category	Severity	Location	Status
Tokenomics	● Low	UltimateFantoms.sol: 1820	Acknowledged

Description

The owner address has permissions to arbitrarily setMintSize which can drastically affect the tokenomics of the project.

Recommendation

Remove the setMintSize function or appropriately timelock it for community trust and safety.

Resolution

Ultimate Fantoms: Acknowledged, contract ownership will be changed to the multisig at

0x87f385d152944689f92Ed523e9e5E9Bd58Ea62ef.

UF-12 | Unequal Minting Rewards

Category	Severity	Location	Status
Logical Error	● Low	UltimateFantoms.sol: 1723	Resolved

Description

In `publicMint` the rewards distribution to `CYBERs` holders can only occur on the first mint and never after.

Recommendation

Ensure this is the expected behavior. If it isn't, refactor the reward logic to more fairly include `CYBERs` holders.

Resolution

Ultimate Fantoms: Resolved.

UF-13 | Mutability Modifiers

Category	Severity	Location	Status
Mutability	● Low	UltimateFantoms.sol	Resolved

Description

The _beetsAlloc, _dfyAlloc, _treasuryAlloc, mintFees, and _earnAmount variables are never modified, and should therefore be declared constant.

Recommendation

Declare them as constant.

Resolution

Ultimate Fantoms: Resolved, applied suggestion where appropriate.

UF-14 | Function Visibility Modifiers

Category	Severity	Location	Status
Optimization	● Low	UltimateFantoms.sol	Resolved

Description

The functions `setRoyaltyAddress`, `updateSpiritRouter`, `updatePaintRouter`, `publicMint`, `setMintFees`, `enableMinting`, `disableMinting`, `setBaseURI`, `setTeamMinting`, `setMintSize`, and `sweepEthToAddress` are marked as `public`, but are never called from inside the contract.

Recommendation

These functions can be marked `external` for gas optimization.

Resolution

Ultimate Fantoms: Resolved, applied suggestion where appropriate.

RS-1 | Unnecessary Code

Category	Severity	Location	Status
Best Practices	● Low	RoyaltySplitter.sol	Resolved

Description

The `_earnAmount` variable goes unused.

Recommendation

Remove unused code.

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

RS-2 | Unequal Royalty Rewards

Category	Severity	Location	Status
Logical Error	● Low	RoyaltySplitter.sol	Resolved

Description

In the recieve and fallback functions, the rewards distribution to CYBERs holders can only occur before the second mint and never after.

Recommendation

Ensure this is the expected behavior. If it isn't, refactor the reward logic to more fairly include CYBERs holders.

Resolution

Ultimate Fantoms: Resolved.

RS-3 | Repetitive Function Calls

Category	Severity	Location	Status
Optimization	● Low	RoyaltySplitter.sol	Resolved

Description

In the `receive` and `fallback` functions the `random` function is called several times, even though the random value is constant during each tx.

Recommendation

Compute the random value once.

Resolution

Ultimate Fantoms: Resolved, applied suggestion.

RS-4 | Mutability Modifiers

Category	Severity	Location	Status
Mutability	● Low	RoyaltySplitter.sol	Resolved

Description

The SPIRITSWAP_ROUTER and _earnAmount variables are never modified, and should therefore be declared constant.

Recommendation

Declare them as constant.

Resolution

Ultimate Fantoms: Resolved, applied suggestion where appropriate.

Auditor’s Verdict

After a line by line manual analysis and automated review, Guardian Audits has concluded that:

- Ultimate Fantom’s smart contracts have a LOW RISK SEVERITY
- Ultimate Fantom’s smart contracts have an **ACTIVE OWNERSHIP**
- Important owner privileges – `setRoyaltyAddress`, `updateSpiritRouter`, `updatePaintRouter`, `setMintSize`, `sweepEthToAddress`

  Ultimate Fantom’s smart contract owner has multiple “write” privileges. Centralization risk correlated to the active ownership is **MEDIUM**

Disclaimer

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 Guardian 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. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’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 Guardian 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.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits