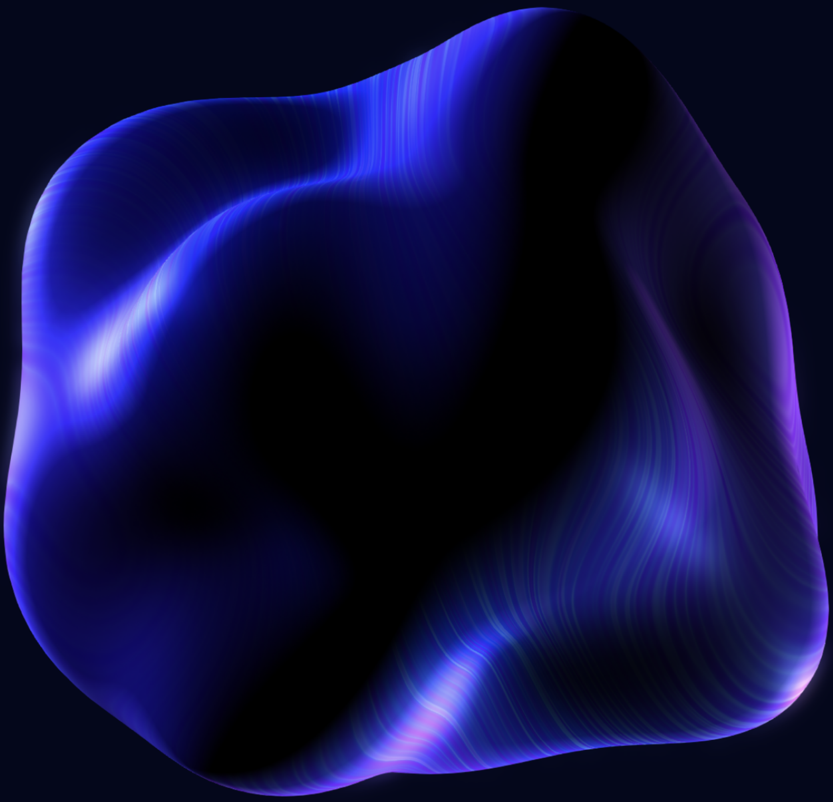
MUNDUS
SECURITY

Security
Smart Contract Audit
Isotopic



ISOTOPIC

mundus.dev   @mundus_security   Mundus.dev

MUNDUS
SECURITY

Isotopic security audit

Reference information

Name	Isotopic
Language	Solidity
Website	https://isotopic.io/
Documentation	https://isotopic.io/whitepaper/
Repository	https://github.com/IsotopicIO/smart-contracts
Initial audit commit	a846f0d7d0d3040699b7cad2532a3e0136983dea
Fix commits	f2779b34c561a9932375827afea09de678d62b40 1e39431f91542d228c7718c8ec92a10dfae2d384 178aff96239219184f1cae9601d8b39b6e7a54c9

Findings summary

Findings statistics

Severity	**Number**	**Left acknowledged**
High	2	0
Medium	2	0
Low	3	1
Informational	7	2
Gas	6	5
Total	**20**	**8**

Finding Severity breakdown

All vulnerabilities discovered during the source code audit are classified based on their potential severity and have the following classification:

Severity	**Description**
High	Bugs that can trigger a contract failure or theft of assets. Further recovery is possible only by manual modification of the contract state or replacement of the contract.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss of funds.
Low	Bugs that do not pose significant danger to the project or its users but are recommended to be fixed nonetheless.
Informational	All other non-essential recommendations.
Gas	Gas optimization recommendations.

Project description

Isotopic is a software distribution service for video game publishers and gamers. Ownership of a game copy in web2 translates into ownership of an NFT in web3.

The present audit report focuses on early-stage **UntransferableGameLicense** contract, which is an extension of ERC721.

Scope of work

Path
smart-contracts/UntransferableGame721/UntransferableGame721.sol

Findings

ID	Severity	Description	Status
01	High	Possible loss of funds due to owner's key compromise	fixed
02	High	Undermined protocol transparency due to publisher change by owner	fixed
03	Medium	mint does not allow for fee-on-transfer tokens	fixed
04	Medium	Unchecked ERC20 transfers in withdraw	fixed
05	Low	Miscalculated owner's fee leaves dust on contract's balance	ack.
06	Low	Missing zero check in setPublisher	fixed
07	Low	ETH locked due to lack of withdrawal logic	fixed
08	Informational	Setter functions should emit events	fixed
09	Informational	Owner fee should be declared constant with more decimals	fixed
10	Informational	Excessive boolean comparisons in mint	fixed
11	Informational	Public access controlled functions should be external	ack.
12	Informational	Unused arguments' names can be omitted in transfer functions	fixed
13	Informational	Excessive _transfer internal override	ack.
14	Informational	Excessive state variables initialization	fixed
15	Gas	Introduce custom errors	ack.
16	Gas	Switch memory to calldata in setBaseURI and setBaseExtension	ack.
17	Gas	Access controlled functions can be made payable	ack.
18	Gas	Use unchecked arithmetics in withdraw	ack.
19	Gas	Whitelist checks order induces gas overhead for public games	fixed
20	Gas	maxSupply and token should be immutable	ack.

Source code audit

ID-01. High: Possible loss of funds due to owner's key compromise

Description

The UntransferableGameLicense contract's governance is centralized at a single owner's address. In the event of owner's private key compromise the access to the game sales revenue as well as the protocol fee will be lost.

Recommendation

Ensure that the owner of the UntransferableGameLicense contract is a Multisig contract.

Alleviation

The Isotopic team explicitly stated that the ownership of a contract will be granted to a Multisig wallet.

ID-02. High: Undermined protocol transparency due to publisher change by owner

Description

The setPublisher function (L1967) of the UntransferableGameLicense contract can be invoked by either the contract's owner of the game publisher. This means that the owner can potentially deprive the game publisher of all sales revenue by changing the publisher state variable to an arbitrary address, thus undermining transparency of the Isotopic protocol.

Recommendation

Modify the setPublisher function (L1967) so that it can only be invoked by the game publisher.

Alleviation

As of commit 178aff96239219184f1cae9601d8b39b6e7a54c9, the setPublisher function can only be called by the game publisher.

ID-03. **Medium**: mint does not allow for fee-on-transfer tokens

Description

The mint function (L1893) of the UntransferableGameLicense contract takes _mintAmount argument and processes a user's game purchase in the following (simplified) way

1. Calculate the total price as cost * _mintAmount (L1905)
2. Transfer the corresponding amount of token from user to the contract (L1905)

The approach described above assumes that the cost * _mintAmount transferred to a contract is equal to the actual received amount of token. This is not the case for fee-on-transfer tokens. Some tokens (e.g. USDT, USDC) do not currently charge a fee but may do so in the future. As a result, the amount of funds transferred to the contract will not be equal to the amount paid by user.

Recommendation

Change the mint function to have a general flow as described below

1. Take tokenAmount as the third argument
2. Compute token balance of the UntransferableGameLicense contract before token transfer
3. Perform transferFrom(,,tokenAmount)
4. Compute token balance of the UntransferableGameLicense contract after token transfer
5. Compute balance diff between steps 4 and 2
6. Check for diff >= cost * _mintAmount

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the mint function implements logic described in the Recommendation section above.

ID-04. **Medium**: Unchecked ERC20 transfers in withdraw

Description

The withdraw function (L1987) of the UntransferableGameLicense contract does not perform checks for successful ERC20 transfers of protocol fee and sales revenue. This can cause errors in intended protocol logic when using an arbitrary ERC20 as a payment token.

Recommendation

Add `require(transfer(...));` to the `withdraw` function of the UntransferableGameLicense contract.

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the `withdraw` function performs additional checks on ERC20 transfers.

ID-05. Low: Miscalculated owner's fee leaves dust on contract's balance

Description

The withdraw function (L1987) of the UntransferableGameLicense contract performs integer division twice to calculate the protocol fee and publisher's sales revenue. This approach leads to ERC20 dust being left each time the `withdraw` function is invoked.

Recommendation

Modify the `withdraw` function to calculate the owner and publisher shares in the following way

```
fee = amount*8/100;
revenue = amount - fee;
```

Alleviation

The Isotopic team indicated, that the dust left after withdrawal is not an issue, and that even split of the sales revenue among the game publisher and owner is more desireable.

ID-06. Low: Missing zero check in `setPublisher`

Description

The `setPublisher` function (L1967) of the UntransferableGameLicense contract performs no check for `_publisher = 0x00` value.

Recommendation

Add `require(_publisher != address(0));` to the `setPublisher` function.

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the `setPublisher` function performs additional zero check for the `_publisher` variable.

ID-07. Low: ETH locked due to lack of withdrawal logic

Description

The `mint` function (L1893) of the UntransferableGameLicense contract is declared `payable`. However, there is no mechanism for ETH withdrawal present in the contract which leads to funds locked in the event of accidental ETH transfer.

Recommendation

Add ETH withdrawal logic to the UntransferableGameLicense contract or remove `payable` modifier of the `mint` function.

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the mint function is no longer payable.

ID-08. Informational: Setter functions should emit events

Description

The following setter functions should emit events for better protocol transparency

- setCost (L1939)
- setBaseURI (L1943)
- setBaseExtension (L1947)
- pause (L1951)
- setOnlyWhitelist (L1955)
- setPublisher (L1967)

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the UntransferableGameLicense contract emits events when any of the functions listed above is invoked.

ID-09. Informational: Owner fee should be declared constant with more decimals

Description

The protocol fee should be declared constant at the beginning of the contract for better readability. Additionally, consider increasing the fee decimals if the fee is to be increased by fractions of percent.

Recommendation

Modify the UntransferableGameLicense contract in the following way (the code snippet also contains the recommendation from issue ID-04)

```
uint16 public constant FEE_PCT = 800;
uint16 public constant BASE_PCT = 10000;

...

function withdraw() public onlyOwnerOrPublisher {
    ...
    uint256 fee = amount*FEE_PCT/BASE_PCT;
    uint256 revenue = amount - fee;
    ...
}
```

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the owner and publisher shares are declared constant with total sum having 4 decimals.

ID-10. Informational: Excessive boolean comparisons in `mint`

Description

The `mint` function (L1893) of the UntransferableGameLicense contract contains excessive bool comparisons to constants

- `if(whitelisted[_to] == false)`
- `require(onlyWhiteList==false, ...);`

Alleviation

As of commit 1e39431f91542d228c7718c8ec92a10dfae2d384, the `mint` function does not contain excessive boolean comparisons.

ID-11. Informational: Public access controlled functions should be external

Description

Public access controlled function should be declared external as hey are not intended to be called from within the contract.

Alleviation

The Isotopic team acknowledges this recommendation.

ID-12. Informational: Unused arguments' names can be omitted in `transfer` functions

Description

The unused arguments' names can be omitted in the following functions

- `transferFrom` (L1975)
- `safeTransferFrom` (L1979)
- `safeTransferFrom` (L1983)

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the unused arguments' names are removed from the overridden `transfer` functions.

ID-13. Informational: Excessive `_transfer internal` override

Description

The UntransferableGameLicense contract contains excessive override of the `_transfer internal function` (L1971).

Alleviation

The Isotopic team acknowledges this recommendation.

ID-14. Informational: Excessive state variables initialization

Description

The following state variables have excessive hard-coded initialization

- `paused` (L1855)
- `onlyWhiteList` (L1856)

Alleviation

As of commit f2779b34c561a9932375827afea09de678d62b40, the hard-coded state variable initialization is removed from the UntransferableGameLicense contract.

ID-15. Gas: Introduce custom errors

Description

Introduce custom errors instead of revert strings to save gas during deployment at at runtime.

Alleviation

The Isotopic team acknowledges this recommendation.

ID-16. Gas: Switch memory to calldata in `setBaseURI` and `setBaseExtension`

Description

Consider switching data location from memory to calldata in the following setter functions for gas savings

- `setBaseURI` (L1943)
- `setBaseExtension` (L1947)

Alleviation

The Isotopic team acknowledges this recommendation.

ID-17. Gas: Access controlled functions can be made payable

Description

If a function is access controlled, e.g. with a `onlyOwner` modifier, the function will revert if a game buyer tries to send ETH by calling it. Marking the function as `payable` will lower the gas cost for the contract's owner and the game publisher because the compiler will not include checks for whether a payment was provided.

Alleviation

The Isotopic team acknowledges this recommendation.

ID-18. Gas: Use unchecked arithmetics in withdraw

Description

Use unchecked arithmetics in the cases where overflow is not possible, such as

- for-loop in _mint (L1911)

```
unchecked{
    for (uint256 i = 1; i <= _mintAmount; i++) {
      _safeMint(_to, supply + i);
    }

    mintedBalance[_to] += _mintAmount;
}
```

- fee calculation in withdraw function (L1987). The following code snippet also contains recommendations from issues ID-04 and ID-09.

```
unchecked {
    uint256 fee = amount*FEE_PCT/BASE_PCT;
    uint256 revenue = amount - fee;
}
```

Alleviation

The Isotopic team acknowledges this recommendation.

ID-19. Gas: Whitelist checks order induces gas overhead for public games

Description

The mint function (L1893) of the UntransferableGameLicense contract implements the whitelist logic in the following way

1. Check if a user is not in the whitelist (L1900)
2. Check if access to the game is public (L1901)

In this setup, if onlyWhiteList is not enabled, step 1 is superfluous and causes unnecessary gas expenses.

Recommendation

Switch the order of the whitelist checks. The code snippet below also contains recommendations from issue ID-10

```
if(onlyWhiteList) {
    require(whitelisted[_to], "only whitelist addresses can mint");
}
```

Alleviation

As of commit 1e39431f91542d228c7718c8ec92a10dfae2d384, the mint function implements the whitelist check in accordance with recommendation above.

ID-20. Gas: maxSupply and token should be immutable

Description

The maxSupply and token state variables should be declared immutable to save gas each time mint or withdraw is called.

Alleviation

The Isotopic team acknowledges this recommendation.

Disclaimers

Mundus disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

Technical disclaimers

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have vulnerabilities that can lead to hacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.