

Playbook Smart Contract Audit

26th October 2021

2nd Review

Summary

This report has been prepared for **Playbook** to discover issues and vulnerabilities in the source code of the **Playbook** 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 Manual Review 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.


Overview


Project Name	Playbook
Platform	Ethereum
Language	Solidity
Codebase / Sourcecode	https://rinkeby.etherscan.io/address/0x18020420635369e9e9febe4f7467e2e4ff0f232e#code


Vulnerability Summary


Vulnerability Level	Total	Pending	Declined	Acknowledged	Resolved
 Critical	0	0	0	0	0
 Major	3	1	0	2	0
 Medium	1	0	0	1	0
 Minor	1	0	0	1	0
 Discussion	6	0	0	6	0

 Critical - Issues that can destroy the project if not resolved with a very high risk

 Major - Issues that can destroy the project if not resolved with a high risk

 Medium - Issues that can set back the project with a high risk. Attackers do not need a high level of technical knowledge to carry out the exploit

 Minor - Issues that can set back the project with a low risk. Attackers do not need a high level of technical knowledge to carry out the exploit

 Discussion - Discussion required to further evaluate the potential severity OR Suggested improvements

Findings

ID	Title	Category	Severity	Status
PB-01	Centralization Risk	Centralization / Privilege	🟡 Major	Pending
PB-02	Repeat Minting Vulnerability	Attack Vulnerability	🟡 Major	Pending
PB-03	Absence of minting window	Attack Vulnerability	🟡 Major	Pending
PB-04	Exposed Base URI	Attack Vulnerability	🟡 Medium	Pending
PB-05	Signer private key	Attack Vulnerability	🟢 Minor	Pending
PB-06	Function to set private mint price	Suggestion	💬 Discussion	Pending
PB-07	Ambiguous use case for WhitelistMint and WhitelistMintSolo	Clarification	💬 Discussion	Pending
PB-08	Unused function recoverSigner	Gas Optimization	💬 Discussion	Pending
PB-09	To use OpenZeppelin's ECDSA library	Suggestion	💬 Discussion	Pending
PB-10	Redundant getPrice function	Gas Optimization	💬 Discussion	Pending
PB-11	Lack of comments	Suggestion	💬 Discussion	Pending

PB-01: Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	Line 1264:1444	Acknowledged

Description

In the contract PB.sol, the owner Role has the authority over the following functions:

- setPrice
- setBaseURI
- pause
- withdrawAll

Any compromise to the owner account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the Owner account's private key to avoid any potential risks of being hacked.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets (Gnosis Safe).

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Resolution

The client has decided to transfer the ownership of the smart contract to a multisignature wallet (gnosis safe) to handle all the admin functions as stated above. During the mint period, the contract owner will still continue to be a centralized wallet.

PB-02: Repeat Minting Vulnerability (From Smart Contracts)

Category	Severity	Location	Status
Attack Vulnerability	🟡 Major	Line 1297:1315	Pending

Description

The mint function may be called by a repeatedly smart contract to bypass the mint limit per wallet.

For example the mint function can be called in a loop.

Users are also able to transfer out their minted NFTs to bypass the maximum NFTs per wallet limit.

Recommendation

We advise the client to implement an onlyEOA modifier to the mint function.

```
modifier onlyEOA() {  
    require(msg.sender == tx.origin, "PB: Only EOA");  
    _;  
}
```

PB-03: Absence of Minting Window

Category	Severity	Location	Status
Attack Vulnerability	● Major	Line 1297:1315	Acknowledged

Description

The mint function may be called as soon as the contract is deployed.

Recommendation

We advise the client to implement a minting window or implement a safeguard to prevent premature mints.

Resolution

The user will start the contract in a paused state (line 1281) to prevent public mint on contract deployment.

PB-04: Exposed Base URI

Category	Severity	Location	Status
Attack Vulnerability	● Medium	Line 1293:1295	Acknowledged

Description

The BaseURI for the NFT metadata is exposed to the public. This allows anyone who has the BaseURI to crawl the metadata and potentially cause metadata leaks.


Recommendation

We advise the client to implement a reveal mechanism which only updates the actual baseURI after all NFTs are minted or after a fixed time window has passed.

Resolution

The client will use the setURI Function to manually reveal the metadata.

PB-05: Signer Private Key

Category	Severity	Location	Status
Attack Vulnerability	 Minor		Acknowledged

Description

The private key of the owner account has the authority to give access to the following functions:

- whitelistMint
- whitelistMintSolo

Any compromise to the private key will allow the attacker to access these functions.

Recommendation

We advise the client to ensure that the private key used to sign the transaction messages are safely secured in their backend services with the best practices.

Resolution

The client ensures that the private keys are stored safely while signing the whitelist transactions. The private key is not used on any backend service.

PB-06: Function to set Whitelist Mint Price

Category	Severity	Location	Status
Suggestion	💬 Discussion		Acknowledged

Description

The client may want to consider having a setter function for the whitelist mint price, as they do with the public mint. In the current edition of the smart contract, the whitelist price is the same as the public price (0.06 ETH).

PB-07: Ambiguous use case for whitelistMint and whitelistMintSolo

Category	Severity	Location	Status
Clarification	🗨 Discussion	Line 1316:1363	Acknowledged

Description

The 2 functions, whitelistMint and whitelistMintSolo are unclear to users when they need to call them in the contract.

Recommendation

We advise the client to provide more details for the difference between these 2 mints.

PB-08: Unused function recoverSigner

Category	Severity	Location	Status
Gas Optimization	💬 Discussion	Line 1416:1422	Acknowledged

Description

The private recoverSigner function was not used in the PB.sol contract.

Recommendation

We advise the client to remove functions that are not used to save on deployment cost.

Resolution

The client will remove the recoverSigner function.

PB-09: To use OpenZeppelin's ECDSA library

Category	Severity	Location	Status
Suggestion	🗨 Discussion	Where ecrecover() is used	Acknowledged

Description

The function ecrecover has a few security flaws:

- In some cases ecrecover can return a random address instead of 0 for an invalid signature.
- Signature are malleable, meaning you might be able to create a second also valid signature for the same data. In our case we are not using the signature data itself (which one may do as an id for example).
- An attacker can construct a hash and signature that look valid if the hash is not computed within the contract itself.

Recommendation

We advise the client to utilize the OpenZeppelin's ECDSA library to extract the signer address to verify the transaction.

PB-10: Redundant getPrice function

Category	Severity	Location	Status
GasOptimization	Discussion	Line 1381:1383	Acknowledged

Description

The function getPrice is redundant as _price is a public state variable. _price is accessible via _price() function.


Recommendation

We advise the client to remove the getPrice function from PB.sol.

Resolution

The client will remove the getPrice function.

PB-11: Lack of comments

Category	Severity	Location	Status
Suggestion	 Discussion	PB.sol	Acknowledged

Description

The PB.sol smart contract should have comments using the NatSpec format as it would be publically available for anyone to read.

Recommendation

We advise the client to add comments for every function using the NatSpec format:

<https://docs.soliditylang.org/en/latest/natspec-format.html>