BLOCKHAT
SECURITY

BlockPark

Smart Contract Security Audit

Prepared by BlockHat

April 13th, 2023 - April 15th, 2023

BlockHat.io

contact@blockhat.io

Document Properties

Client	BlockPark
Version	0.1
Classification	Public

Scope

The BlockPark Contract in the BlockPark Repository

Link	Address
https://bscscan.com/address/0xEc202b99b5ac4c48f3864cF159369ed1368B62DA#code	0xEc202b99b5ac4c48f3864cF159369ed1368B62DA

Files	MD5 Hash
/PropToken.sol	caa18a9d87cd7fe81608331e1ad06923

Contacts

COMPANY	CONTACT
BlockHat	contact@blockhat.io

Contents

1 Introduction

BlockPark engaged BlockHat to conduct a security assessment on the BlockPark beginning on April 13th, 2023 and ending April 15th, 2023. In this report, we detail our methodical approach to evaluate potential security issues associated with the implementation of smart contracts, by exposing possible semantic discrepancies between the smart contract code and design document, and by recommending additional ideas to optimize the existing code. Our findings indicate that the current version of smart contracts can still be enhanced further due to the presence of many security and performance concerns.

This document summarizes the findings of our audit.

1.1 About BlockPark

BlockPark is a solution based real estate investment platform tokenizing real estate ownership for buyers and sellers interested in building wealth while earning passive income. Users are given tools to create and manage their assets using BlockPark property management software and incentivized to grow their portfolio using PROP tokens.

Issuer	BlockPark
Website	`https://theblockpark.com/`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

BlockHat used a combination of manual and automated security testing to achieve a balance between efficiency, timeliness, practicability, and correctness within the audit's scope. While manual testing is advised for identifying problems in logic, procedure, and implementation, automated testing techniques help to expand the coverage of smart contracts and can quickly detect code that does not comply with security best practices.

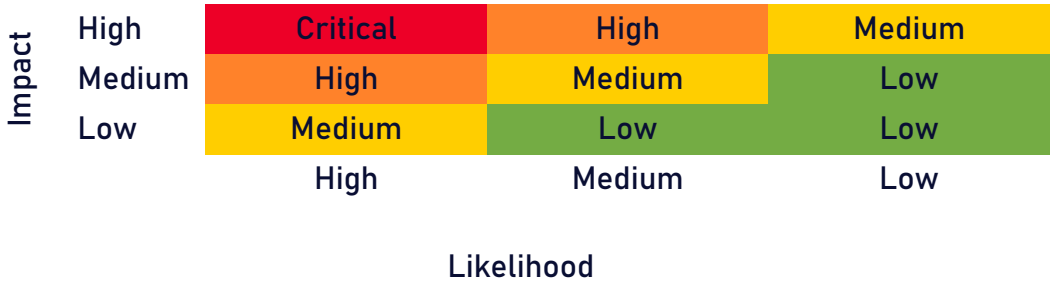
1.2.1 Risk Methodology

Vulnerabilities or bugs identified by BlockHat are ranked using a risk assessment technique that considers both the LIKELIHOOD and IMPACT of a security incident. This framework is effective at conveying the features and consequences of technological vulnerabilities.

Its quantitative paradigm enables repeatable and precise measurement, while also revealing the underlying susceptibility characteristics that were used to calculate the Risk scores. A risk level will be assigned to each vulnerability on a scale of 5 to 1, with 5 indicating the greatest possibility or impact.

- Likelihood quantifies the probability of a certain vulnerability being discovered and exploited in the untamed.
- Impact quantifies the technical and economic costs of a successful attack.
- Severity indicates the risk's overall criticality.

Probability and impact are classified into three categories: H, M, and L, which correspond to high, medium, and low, respectively. Severity is determined by probability and impact and is categorized into four levels, namely Critical, High, Medium, and Low.



2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the BlockPark implementation. During the first part of our audit, we examine the smart contract source code and run the codebase via a static code analyzer. The objective here is to find known coding problems statically and then manually check (reject or confirm) issues highlighted by the tool. Additionally, we check business logics, system processes, and DeFi-related components manually to identify potential hazards and/or defects.

2.2 Key Findings

In general, these smart contracts are well-designed and constructed, but their implementation might be improved by addressing the discovered flaws, which include , 3 medium-severity, 6 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
Centralized control over blacklist and whitelist	MEDIUM	Not fixed
Contract ownership transfer	MEDIUM	Not fixed
Inconsistent use of access control	MEDIUM	Not fixed
Unnecessary ownership pattern	LOW	Not fixed
No token recovery mechanism	LOW	Not fixed
Incorrect use of msg.sender	LOW	Not fixed
Token transfers blocked without distinction during pause	LOW	Not fixed
Missing address verification	LOW	Not fixed
Floating Pragma	LOW	Not fixed

3 Finding Details

A PropToken.sol

A.1 Centralized control over blacklist and whitelist [MEDIUM]

Description:

The contract provides the ability for administrators to add or remove addresses from the blacklist and whitelist, which could lead to potential centralization and censorship concerns for users.

Code:

Listing 1: PropToken.sol

```
38      function changeWhitelist(address _user, bool _status)
39          public
40          onlyRole(ADMIN_ROLE)
41      {
42          whitelist[_user] = _status;
43          emit WhitelistChanged(_user, _status);
44      }

46      function changeBlocklist(address _user, bool _status)
47          public
48          onlyRole(ADMIN_ROLE)
49      {
50          blocklist[_user] = _status;
51          emit BlocklistChanged(_user, _status);
52      }
```

Risk Level:

Likelihood – 3
Impact - 3

Recommendation:

Consider implementing a more decentralized approach for managing the blacklist and whitelist, such as using a decentralized governance model where token holders can vote on additions or removals from the lists. Alternatively, assess the necessity of these features and consider removing them if they are not essential for the contract's functionality.

Status - Not fixed

A.2 Contract ownership transfer [MEDIUM]

Description:

The contract allows transferring ownership, but the ADMIN_ROLE is not updated accordingly.

Code:

Listing 2: PropToken.sol

```
69      function setNewOwner(address _newOwner) external {
70          require(
71              owner == _msgSender(),
72              "This function can only be called by the current owner."
73          );
74          emit OwnershipTransfered(owner, _newOwner);
75          owner = _newOwner;
76      }
```

Risk Level:

Likelihood – 3
Impact - 2

Recommendation:

Add a function to remove the ADMIN_ROLE from the old owner and grant it to the new owner.

Status - Not fixed

A.3 Inconsistent use of access control [MEDIUM]

Description:

The setNewOwner function allows only the current owner to change ownership, while other all functions use the ADMIN_ROLE for access control.

Code:

Listing 3: PropToken.sol

```
69      function setNewOwner(address _newOwner) external {
70          require(
71              owner == _msgSender(),
72              "This function can only be called by the current owner."
73          );
74          emit OwnershipTransfered(owner, _newOwner);
75          owner = _newOwner;
76      }
```

Listing 4: PropToken.sol

```
30      function pause() public onlyRole(ADMIN_ROLE) {
31          _pause();
32      }
```

Listing 5: PropToken.sol

```
34      function unpause() public onlyRole(ADMIN_ROLE) {
35          _unpause();
36      }
```

Listing 6: PropToken.sol

```
38      function changeWhitelist(address _user, bool _status)
39          public
40          onlyRole(ADMIN_ROLE)
41      {
42          whitelist[_user] = _status;
43          emit WhitelistChanged(_user, _status);
44      }
```

Listing 7: PropToken.sol

```
46      function changeBlocklist(address _user, bool _status)
47          public
48          onlyRole(ADMIN_ROLE)
49      {
50          blocklist[_user] = _status;
51          emit BlocklistChanged(_user, _status);
52      }
```

Risk Level:

Likelihood – 4
Impact - 2

Recommendation:

Modify the setNewOwner function to use the ADMIN_ROLE for access control, ensuring consistent access control across administrative functions.

Status - Not fixed

A.4 Unnecessary ownership pattern [LOW]

Description:

The contract implements a custom ownership pattern, but it could rely on AccessControl's built-in functionality.

Code:

Listing 8: PropToken.sol

```
11      address private owner;
```

Listing 9: PropToken.sol

```
27       owner = msg.sender;
```

Listing 10: PropToken.sol

```
69      function setNewOwner(address _newOwner) external {
70          require(
71              owner == _msgSender(),
72              "This function can only be called by the current owner."
73          );
74          emit OwnershipTransfered(owner, _newOwner);
75          owner = _newOwner;
76      }
```

Listing 11: PropToken.sol

```
82       function getOwner() public view returns (address) {
83          return owner;
84      } owner = msg.sender;
```

Risk Level:

Likelihood – 2
Impact - 1

Recommendation:

Remove the custom ownership pattern and rely on AccessControl's built-in functionality for managing roles and access control.

Status - Not fixed

A.5 No token recovery mechanism [LOW]

Description:

The contract does not include a mechanism to recover tokens accidentally sent to the contract address.

Code:

Listing 12: PropToken.sol

```
8  contract PropToken is ERC20, Pausable, AccessControl {
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Add a token recovery function that allows the contract owner or an administrator to recover tokens accidentally sent to the contract address.

Status - Not fixed

A.6 Incorrect use of msg.sender [LOW]

Description:

The _beforeTokenTransfer function uses msg.sender instead of _from to check for whitelist status.

Code:

Listing 13: PropToken.sol

```
54      function _beforeTokenTransfer(
55          address _from,
56          address _to,
57          uint256 _amount
58      ) internal override {
59          require(
60              !blocklist[_from] && !blocklist[_to],
61              "Address in the blocklisted."
62          );
63          if (paused()) {
64              require(whitelist[msg.sender], "Token on pause.");
65          }
66          super._beforeTokenTransfer(_from, _to, _amount);
67      }
```

Risk Level:

Likelihood – 2

Impact - 2

Recommendation:

Replace msg.sender with _from to correctly check the address initiating the token transfer during a paused state.

Status - Not fixed

A.7 Token transfers blocked without distinction during pause [LOW]

Description:

When the contract is paused, all token transfers are blocked except for those initiated by whitelisted addresses. This may cause potential disruptions for regular users who are not whitelisted.

Code:

Listing 14: PropToken.sol

```
54      function _beforeTokenTransfer(
55          address _from,
56          address _to,
57          uint256 _amount
58      ) internal override {
59          require(
60              !blocklist[_from] && !blocklist[_to],
61              "Address in the blocklisted."
62          );
63          if (paused()) {
64              require(whitelist[msg.sender], "Token on pause.");
65          }
66          super._beforeTokenTransfer(_from, _to, _amount);
67      }
```

Risk Level:

Likelihood – 2
Impact - 2

Recommendation:

Evaluate the intended use case for the pause function and consider allowing specific types of transfers to continue during the paused state if required. For example, if the purpose of the pause function is to halt trading in case of an emergency, it may be worth allowing transfers between regular users while blocking transfers to and from exchanges.

Status - Not fixed

A.8 Missing address verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type argument changeWhitelist , changeBlocklist and setNewOwner function should include a zero-address test for the address _user and address _newOwner

Code:

Listing 15: PropToken.sol

```
38        function changeWhitelist(address _user, bool _status)
39         public
40         onlyRole(ADMIN_ROLE)
41     {
42         whitelist[_user] = _status;
43         emit WhitelistChanged(_user, _status);
44     }

46     function changeBlocklist(address _user, bool _status)
47         public
```

```
48          onlyRole(ADMIN_ROLE)
49      {
50          blocklist[_user] = _status;
51          emit BlocklistChanged(_user, _status);
52      }
```

Listing 16: PropToken.sol

```
69      function setNewOwner(address _newOwner) external {
70          require(
71              owner == _msgSender(),
72              "This function can only be called by the current owner."
73          );
74          emit OwnershipTransfered(owner, _newOwner);
75          owner = _newOwner;
76      }
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

It is recommended to verify that the address provided in the arguments is different from the address(0) .

Status - Not fixed

A.9 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.17. Contracts should be deployed using the same compiler version and flags that were used during the testing process. Locking the pragma helps ensuring that contracts are not unintentionally deployed using an-

other pragma, such as an obsolete version that may introduce issues in the contract system.

Code:

Listing 17: PropToken.sol

```
2 pragma solidity ^0.8.17;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

Consider locking the pragma version. It is advised that floating pragma not be used in production.Both truffle-config.js and hardhat.config.js support locking the pragma version.

Status - Not fixed

4 Static Analysis (Slither)

Description:

Block Hat expanded the coverage of the specific contract areas using automated testing methodologies. Slither, a Solidity static analysis framework, was one of the tools used. Slither was run on all-scoped contracts in both text and binary formats. This tool can be used to test mathematical relationships between Solidity instances statically and variables that allow for the detection of errors or inconsistent usage of the contracts' APIs throughout the entire codebase.

Results:

```
PropToken.setNewOwner(address)._newOwner (PropToken.sol#69) lacks a zero
   ↪ -check on :
            - owner = _newOwner (PropToken.sol#75)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #missing-zero-address-validation

Different versions of Solidity are used:
       - Version used: ['^0.8.0', '^0.8.17']
       - ^0.8.17 (PropToken.sol#2)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/access/
          ↪ AccessControl.sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/access/
          ↪ IAccessControl.sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/security/Pausable.
          ↪ sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/ERC20.
          ↪ sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/IERC20
          ↪ .sol#4)
       - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
          ↪ extensions/IERC20Metadata.sol#4)
```

```
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings.sol
          ↪ #4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/ERC165.sol#4)
        - ^0.8.0 (node_modules/@openzeppelin/contracts/utils/
          ↪ introspection/IERC165.sol#4)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #different-pragma-directives-are-used

AccessControl._setRoleAdmin(bytes32,bytes32) (node_modules/@openzeppelin
   ↪ /contracts/access/AccessControl.sol#214-218) is never used and
   ↪ should be removed
AccessControl._setupRole(bytes32,address) (node_modules/@openzeppelin/
   ↪ contracts/access/AccessControl.sol#205-207) is never used and
   ↪ should be removed
Context._msgData() (node_modules/@openzeppelin/contracts/utils/Context.
   ↪ sol#21-23) is never used and should be removed
ERC20._burn(address,uint256) (node_modules/@openzeppelin/contracts/token
   ↪ /ERC20/ERC20.sol#280-295) is never used and should be removed
Strings.toHexString(address) (node_modules/@openzeppelin/contracts/utils
   ↪ /Strings.sol#72-74) is never used and should be removed
Strings.toHexString(uint256) (node_modules/@openzeppelin/contracts/utils
   ↪ /Strings.sol#41-52) is never used and should be removed
Strings.toString(uint256) (node_modules/@openzeppelin/contracts/utils/
   ↪ Strings.sol#16-36) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #dead-code

Pragma version^0.8.17 (PropToken.sol#2) necessitates a version too
   ↪ recent to be trusted. Consider deploying with 0.6.12/0.7.6/0.8.7
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
   ↪ AccessControl.sol#4) allows old versions
```

```
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
   ↪ IAccessControl.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/
   ↪ Pausable.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
   ↪ ERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
   ↪ IERC20.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
   ↪ extensions/IERC20Metadata.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context
   ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings
   ↪ .sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
   ↪ introspection/ERC165.sol#4) allows old versions
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
   ↪ introspection/IERC165.sol#4) allows old versions
solc-0.8.17 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #incorrect-versions-of-solidity

Parameter PropToken.changeWhitelist(address,bool)._user (PropToken.sol
   ↪ #38) is not in mixedCase
Parameter PropToken.changeWhitelist(address,bool)._status (PropToken.sol
   ↪ #38) is not in mixedCase
Parameter PropToken.changeBlocklist(address,bool)._user (PropToken.sol
   ↪ #46) is not in mixedCase
Parameter PropToken.changeBlocklist(address,bool)._status (PropToken.sol
   ↪ #46) is not in mixedCase
Parameter PropToken.setNewOwner(address)._newOwner (PropToken.sol#69) is
   ↪  not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #conformance-to-solidity-naming-conventions
```

```
pause() should be declared external:
	- PropToken.pause() (PropToken.sol#30-32)
unpause() should be declared external:
	- PropToken.unpause() (PropToken.sol#34-36)
changeWhitelist(address,bool) should be declared external:
	- PropToken.changeWhitelist(address,bool) (PropToken.sol#38-44)
changeBlocklist(address,bool) should be declared external:
	- PropToken.changeBlocklist(address,bool) (PropToken.sol#46-52)
decimals() should be declared external:
	- ERC20.decimals() (node_modules/@openzeppelin/contracts/token/
	  ↪ ERC20/ERC20.sol#87-89)
	- PropToken.decimals() (PropToken.sol#78-80)
getOwner() should be declared external:
	- PropToken.getOwner() (PropToken.sol#82-84)
grantRole(bytes32,address) should be declared external:
	- AccessControl.grantRole(bytes32,address) (node_modules/
	  ↪ @openzeppelin/contracts/access/AccessControl.sol#144-146)
revokeRole(bytes32,address) should be declared external:
	- AccessControl.revokeRole(bytes32,address) (node_modules/
	  ↪ @openzeppelin/contracts/access/AccessControl.sol#159-161)
renounceRole(bytes32,address) should be declared external:
	- AccessControl.renounceRole(bytes32,address) (node_modules/
	  ↪ @openzeppelin/contracts/access/AccessControl.sol#179-183)
name() should be declared external:
	- ERC20.name() (node_modules/@openzeppelin/contracts/token/ERC20/
	  ↪ ERC20.sol#62-64)
symbol() should be declared external:
	- ERC20.symbol() (node_modules/@openzeppelin/contracts/token/
	  ↪ ERC20/ERC20.sol#70-72)
totalSupply() should be declared external:
	- ERC20.totalSupply() (node_modules/@openzeppelin/contracts/token
	  ↪ /ERC20/ERC20.sol#94-96)
balanceOf(address) should be declared external:
```

```
        - ERC20.balanceOf(address) (node_modules/@openzeppelin/contracts/
            ↪ token/ERC20/ERC20.sol#101-103)
    transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (node_modules/@openzeppelin/
            ↪ contracts/token/ERC20/ERC20.sol#113-117)
    approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (node_modules/@openzeppelin/
            ↪ contracts/token/ERC20/ERC20.sol#136-140)
    transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (node_modules/
            ↪ @openzeppelin/contracts/token/ERC20/ERC20.sol#158-167)
    increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (node_modules/
            ↪ @openzeppelin/contracts/token/ERC20/ERC20.sol#181-185)
    decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (node_modules/
            ↪ @openzeppelin/contracts/token/ERC20/ERC20.sol#201-210)
    Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
        ↪ #public-function-that-could-be-declared-external
    PropToken.sol analyzed (11 contracts with 78 detectors), 44 result(s)
        ↪ found
```

Conclusion:

Most of the vulnerabilities found by the analysis have already been addressed by the smart contract code review. w

5 Conclusion

We examined the design and implementation of BlockPark in this audit and found several issues of various severities. We advise BlockPark team to implement the recommendations contained in all 9 of our findings to further enhance the code's security. It is of utmost priority to start by addressing the most severe exploit discovered by the auditors then followed by the remaining exploits, and finally we will be conducting a re-audit following the implementation of the remediation plan contained in this report.

We would much appreciate any constructive feedback or suggestions regarding our methodology, audit findings, or potential scope gaps in this report.

BLOCKHAT
SECURITY

For a Smart Contract Audit, contact us at contact@blockhat.io