INTELLY

Smart Contract Security Audit

Prepared by ShellBoxes

July 15th, 2022 - July 22nd, 2022

Shellboxes.com

contact@shellboxes.com

Document Properties

Client	Intelly
Version	1.0
Classification	Public

Scope

The INTELLY Contract in the INTELLY Repository

Repo	Commit Hash
https://gitlab.com/intelly-tech/chain	cda581acb31f9017543b3ce89b8f263aa5cd22c3
https://gitlab.com/intelly-tech/chain	920eee1e81f20dc1b7d8d3454bf9e2b3a1226cc5

Files	MD5 Hash
Access.sol	7ca825e49f9d15f3a662b91b32212ec2
Estate.sol	fa392dd83a4d4566f3a9a948c0171033
Exchange.sol	35a1451f69e900f7c4c68bf35be3a338
Oracle.sol	d6be5e4cb55c9ace48fc433abab38908
Trader.sol	37c38b534a1564b501bd24a9a3360efe
local/Stable.sol	f97e1e4c5ffd3cd95c229c0d37bf3114
local/Token.sol	2646f8d50d8616d9599e971dc984ec3e

Re-Audit Files

Files	MD5 Hash
Access.sol	96b5e0a8fa26f924aa885ec68ef97dc9
Investment.sol	d0eccfca97f5ce9e2dbaada63c0bff04
Oracle.sol	1880def0b81a8575aebb8f3f40669bec
Platform.sol	38fe9e5c5b951978fc80c74511b01d06
Stable.sol	1934310b800a4595f016f97850176b82
Swap.sol	6f86bfea71f7880bf92bdf483f3018b5
Token.sol	f5e037f9848496ad6210c1e61aaa074c

Contacts

COMPANY	EMAIL
ShellBoxes	contact@shellboxes.com

Contents

1 Introduction

Intelly engaged ShellBoxes to conduct a security assessment on the INTELLY beginning on July 15th, 2022 and ending July 22nd, 2022. 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 Intelly

Intelly was founded with the inspiration of changing how real estate investment works. Enabling people to benefit from the power of blockchain and opening the world of real estate investment for small size individual investors.

Issuer	Intelly
Website	`https://intelly.tech/`
Type	Solidity Smart Contract
Audit Method	Whitebox

1.2 Approach & Methodology

ShellBoxes 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 ShellBoxes 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.

Impact				
	High	Critical	High	Medium
	Medium	High	Medium	Low
	Low	Medium	Low	Low
		High	Medium	Low

Likelihood

2 Findings Overview

2.1 Summary

The following is a synopsis of our conclusions from our analysis of the INTELLY 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 1 critical-severity, 3 medium-severity, 15 low-severity vulnerabilities.

Vulnerabilities	Severity	Status
The Fixed Price Of Any Amount Below 1000000 is Zero	CRITICAL	Fixed
Fees should be limited	MEDIUM	Fixed
The Operator Can Burn Any Token	MEDIUM	Acknowledged
Missing Transfer Verification	MEDIUM	Fixed
Missing Value Verification	LOW	Fixed
Missing Address Verification	LOW	Fixed
Floating Pragma	LOW	Fixed
Missing Value Verification	LOW	Fixed
Missing Address Verification	LOW	Fixed
Floating Pragma	LOW	Fixed
Missing Address Verification	LOW	Fixed
Floating Pragma	LOW	Fixed
Missing Address Verification	LOW	Fixed
Floating Pragma	LOW	Fixed

Approve Race Condition	LOW	Acknowledged
Floating Pragma	LOW	Fixed
Approve Race Condition	LOW	Acknowledged
Floating Pragma	LOW	Fixed
Floating Pragma	LOW	Fixed

3 Finding Details

A Oracle.sol

A.1 The Fixed Price Of Any Amount Below 1000000 is Zero [CRITICAL]

Description:

When fixed is equal to true, the price can be obtained using the _getFixed function. Due to a type conversion mistake, the price will always be equal to "0" for any amount less than 1000000. This problem will impact all Estate.sol, Exchange.sol, and Trader.sol contracts that depend on the Oracle.sol to determine the token's price.

Code:

Listing 1: Oracle.sol

```
138 function _getFixed(uint amount, address[] memory path)
139     internal
140     view
141     returns (uint)
142 {
143     uint ratio;
144     hasPermit(TOKEN_PERMIT, path[0]) ? ratio = fromRatio : ratio =
            ↪ toRatio;
145     return (amount / MEASURE) * ratio;
146 }
```

Risk Level:

Likelihood – 5
Impact - 5

Recommendation:

It is recommended to perform the multiplication operation before the division, then add a require statement that makes sure that amount*ratio is higher than MEASURE.

Status - Fixed

The Intelly team has solved the issue by performing the multiplication operation before the division, and adding a require statement that verifies that amount * ratio is higher than MEASURE.

A.2 Missing Value Verification [LOW]

Description:

Certain functions lack a value safety check, the values of the arguments should be verified to allow only the ones that comply with the contract's logic. The contract should ensure that, while changing the fromRatio and toRatio, one of those variables is more than MEASURE and the other is lower. Additionally, the length of the path array input in the _getFixed and _getRouted functions should be confirmed to be two.

Code:

Listing 2: Oracle.sol

```
138 function _getFixed(uint amount, address[] memory path)
139     internal
140     view
141     returns (uint)
142 {
143     uint ratio;
144     hasPermit(TOKEN_PERMIT, path[0]) ? ratio = fromRatio : ratio =
        ↪ toRatio;
145     return (amount / MEASURE) * ratio;
146 }
```

Listing 3: Oracle.sol

```
148 function _getRouted(uint amount, address[] memory path)
149     internal
150     view
151     returns (uint)
152 {
153     uint[2] memory amounts;
154     amounts = IRouter(router).getAmountsOut(amount, path);
155     return amounts[1];
156 }
```

Listing 4: Oracle.sol

```
176 function setFromRatio(uint _fromRatio) public onlyRole(OPERATOR_ROLE) {
177     fromRatio = _fromRatio;
178 }
```

Listing 5: Oracle.sol

```
180 function setToRatio(uint _toRatio) public onlyRole(OPERATOR_ROLE) {
181     toRatio = _toRatio;
182 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you verify the values provided in the arguments. The issue can be addressed by utilizing a require statement.

Status - Fixed

The Intelly team has solved the issue by requiring one of the fromRatio and toRatio variables to be more than MEASURE and the other is lower.

A.3 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. The `_access` argument should be verified to be different from the `address(0)`, and the `_router` argument should be verified to be the same as the pancakeswap router address.

Code:

Listing 6: Oracle.sol

```
39 constructor(
40     address _access,
41     address _router,
42     address _token,
43     address _stable
44 ) {
45     access = _access;
46     router = _router;
47     _setupPermit(TOKEN_PERMIT, _token);
48     _setupPermit(EXCHANGE_PERMIT, _token);
49     _setupPermit(EXCHANGE_PERMIT, _stable);
50 }
```

Listing 7: Oracle.sol

```
184 function setAccess(address _access) public onlyRole(OPERATOR_ROLE) {
185     access = _access;
186 }
```

Listing 8: Oracle.sol

```
188 function setRouter(address _router) public onlyRole(OPERATOR_ROLE) {
189     router = _router;
190 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Intelly team has solved the issue by adding require statements to make sure the addresses provided in the arguments are different from the address(0).

A.4 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.7. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 9: Oracle.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

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

Status - Fixed

The Intelly team has solved the issue by locking the pragma version to 0.8.7.

B Estate.sol

B.1 Fees should be limited [MEDIUM]

Description:

The fee variable is modifiable by the operator, the setter does not have a restriction about the values that the fee variable can take. This implies that the operator can raise the price without being constrained, which could result in the contract having astronomical royalty fees.

Code:

Listing 10: Estate.sol

```
38  constructor(
39      address _creator,
40      address _access,
41      address _token,
42      address _stable,
43      address _oracle,
44      uint _amount,
45      uint _fee,
46      uint _limit
47  ) ERC1155("https://a2nnpid9qnmy.usemoralis.com/{id}.json") {
48      creator = _creator;
```

```
49      access = _access;
50      token = _token;
51      stable = _stable;
52      oracle = _oracle;
53      fee = _fee;
54      limit = _limit;
55      path = [stable, token];
56      _mint(_creator, 1, _amount, "");
57  }
```

Listing 11: Estate.sol

```
134 function setFee(uint _fee) public onlyRole(OPERATOR_ROLE) {
135     fee = _fee;
136 }
```

Risk Level:

Likelihood – 2
Impact - 4

Recommendation:

Consider limiting the value that the fee can have in both the constructor and the fee setter.

Status - Fixed

The Intelly team has solved the issue by limiting the value of the fee.

B.2 The Operator Can Burn Any Token [MEDIUM]

Description:

The operator have the ability to burn anyone's token using the burn function, this represents a significant centralization risk where the operator have control over everyone's tokens.

Code:

Listing 12: Estate.sol

```
158 function burn(
159     address from,
160     uint id,
161     uint amount
162 ) public onlyRole(OPERATOR_ROLE) {
163     _burn(from, id, amount);
164 }
```

Risk Level:

Likelihood – 2
Impact - 5

Recommendation:

Consider removing the functionality or restricting it to only allow the holder to burn his own tokens.

Status - Acknowledged

The Intelly team has acknowledged the finding, stating that the functionality cannot be changed for regulatory purposes under the BVI Sıba Regulatory (BVI Securities Investment Business Act 2010) requirements.

B.3 Missing Value Verification [LOW]

Description:

Certain functions lack a value safety check, the values of the arguments should be verified to allow only the ones that comply with the contract's logic. The contract must ensure that `_limit`, `_fee` and `_amount` are different from 0 in both the constructor and the setters. Also, the `_path` argument should be verified to have a length of 2.

Code:

Listing 13: Estate.sol

```
38 constructor(
39     address _creator,
40     address _access,
41     address _token,
42     address _stable,
43     address _oracle,
44     uint _amount,
45     uint _fee,
46     uint _limit
47 ) ERC1155("https://a2nnpid9qnmy.usemoralis.com/{id}.json") {
48     creator = _creator;
49     access = _access;
50     token = _token;
51     stable = _stable;
52     oracle = _oracle;
53     fee = _fee;
54     limit = _limit;
55     path = [stable, token];
56     _mint(_creator, 1, _amount, "");
57 }
```

Listing 14: Estate.sol

```
134 function setFee(uint _fee) public onlyRole(OPERATOR_ROLE) {
135     fee = _fee;
136 }
```

Listing 15: Estate.sol

```
138 function setLimit(uint _limit) public onlyRole(OPERATOR_ROLE) {
139     limit = _limit;
140 }
```

Listing 16: Estate.sol

```
146 function setPath(address[] memory _path) public onlyRole(OPERATOR_ROLE)
        ↪ {
147     path = _path;
148 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you verify the values provided in the arguments. The issue can be addressed by utilizing a require statement.

Status - Fixed

The Intelly team has fixed the issue by verifying the values provided in the arguments to match with the logic of the smart contract.

B.4 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. The _creator, _access, _token, _stable, _oracle, _path[0] and _path[1] arguments should be verified to be different than the address(0).

Code:

Listing 17: Estate.sol

```
38 constructor(
```

```
39      address _creator,
40      address _access,
41      address _token,
42      address _stable,
43      address _oracle,
44      uint _amount,
45      uint _fee,
46      uint _limit
47  ) ERC1155("https://a2nnpid9qnmy.usemoralis.com/{id}.json") {
48      creator = _creator;
49      access = _access;
50      token = _token;
51      stable = _stable;
52      oracle = _oracle;
53      fee = _fee;
54      limit = _limit;
55      path = [stable, token];
56      _mint(_creator, 1, _amount, "");
57  }
```

Listing 18: Estate.sol

```
146 function setPath(address[] memory _path) public onlyRole(OPERATOR_ROLE)
        ↪ {
147     path = _path;
148 }
```

Risk Level:

Likelihood – 1

Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Intelly team has fixed the issue by verifying the addresses provided in the arguments to be different from the address(0).

B.5 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.7. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 19: Estate.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

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

Status - Fixed

The Intelly team has solved the issue by locking the pragma version to 0.8.7.

C Trader.sol

C.1 Missing Transfer Verification [MEDIUM]

Description:

The ERC20 standard token implementation functions return the transaction status as a Boolean. It is a good practice to check for the return status of the function call to ensure that the transaction was executed successfully. It is the developer's responsibility to enclose these function calls with require() to ensure that, when the intended ERC20 function call returns false, the caller transaction also fails.

Code:

Listing 20: Trader.sol

```
142 function purchase(
143     address inft,
144     uint id,
145     uint amount,
146     address[] memory path
147 ) public nonReentrant {
148     _checkRole(USER_ROLE);
149     emit Purchased(inft, _msgSender(), amount);
150     Listing memory item = listings[inft][id];
151     uint total = item.price * amount;
152     uint price = _getPrice(total, path);
153     IToken(token).transferFrom(_msgSender(), item.creator, price);
154     IEstate(inft).safeTransferFrom(
155         item.creator,
156         _msgSender(),
157         id,
158         amount,
159         ""
160     );
161 }
```

Risk Level:

Likelihood – 1
Impact - 4

Recommendation:

Use the safeTransfer function from the safeERC20 Implementation, or put the transfer call inside an assert or require verifying that it returned true.

Status - Fixed

The Intelly team has fixed the issue by adding a require statement to make sure the transfer has passed successfully.

C.2 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. The _admin, _access, _oracle, _stable and _token arguments should be verified to be different from the address(0).

Code:

Listing 21: Trader.sol

```
55 constructor(
56     address _access,
57     address _admin,
58     address _oracle,
59     address _stable,
60     address _token
61 ) {
62     access = _access;
```

```
63      admin = _admin;
64      oracle = _oracle;
65      stable = _stable;
66      token = _token;
67  }
```

Listing 22: Trader.sol

```
163 function setAccess(address _access) public onlyRole(OPERATOR_ROLE) {
164     access = _access;
165 }

167 function setAdmin(address _admin) public onlyRole(OPERATOR_ROLE) {
168     admin = _admin;
169 }

171 function setOracle(address _oracle) public onlyRole(OPERATOR_ROLE) {
172     oracle = _oracle;
173 }

175 function setStable(address _stable) public onlyRole(OPERATOR_ROLE) {
176     stable = _stable;
177 }

179 function setToken(address _token) public onlyRole(OPERATOR_ROLE) {
180     token = _token;
181 }
```

Risk Level:

Likelihood – 1

Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Intelly team has fixed the issue by verifying the addresses provided in the arguments to be different from the address(0).

C.3 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.7. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 23: Trader.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

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

Status - Fixed

The Intelly team has solved the issue by locking the pragma version to 0.8.7.

D Exchange.sol

D.1 Missing Address Verification [LOW]

Description:

Certain functions lack a safety check in the address, the address-type arguments should include a zero-address test, otherwise, the contract's functionality may become inaccessible. The _admin, _access and _oracle arguments should be verified to be different from the address(0).

Code:

Listing 24: Exchange.sol

```
34 constructor(
35     address _admin,
36     address _access,
37     address _oracle
38 ) {
39     admin = _admin;
40     access = _access;
41     oracle = _oracle;
42 }
```

Listing 25: Exchange.sol

```
103 function setAccess(address _access) public onlyRole(OPERATOR_ROLE) {
104     access = _access;
105 }
```

Listing 26: Exchange.sol

```
107 function setOracle(address _oracle) public onlyRole(OPERATOR_ROLE) {
108     oracle = _oracle;
109 }
```

Listing 27: Exchange.sol

```
111 function setAdmin(address _admin) public onlyRole(OPERATOR_ROLE) {
112     admin = _admin;
113 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend that you make sure the addresses provided in the arguments are different from the address(0).

Status - Fixed

The Intelly team has fixed the issue by verifying the addresses provided in the arguments to be different from the address(0).

D.2 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.7. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 28: Exchange.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

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

Status - Fixed

The Intelly team has solved the issue by locking the pragma version to 0.8.7.

E Token.sol

E.1 Approve Race Condition [LOW]

Description:

The standard ERC20 implementation contains a widely known racing condition in its approve function, wherein a spender can witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Code:

Listing 29: Token.sol

```
7  contract Token is ERC20 {
8      constructor() ERC20("Intelly Token", "INTL") {
9          _mint(msg.sender, 900 * 10**18);
10     }
11 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend using increaseAllowance and decreaseAllowance functions to modify the approval amount instead of using the approve function to modify it.

Status - Acknowledged

The Intelly team has acknowledged the risk, stating that the contract will not be deployed.

E.2 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.7. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 30: Token.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

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

Status - Fixed

The Intelly team has solved the issue by locking the pragma version to 0.8.7.

F Stable.sol

F.1 Approve Race Condition [LOW]

Description:

The standard ERC20 implementation contains a widely known racing condition in its approve function, wherein a spender can witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Code:

Listing 31: Stable.sol

```
7  contract Token is ERC20 {
8      constructor() ERC20("Intelly Token", "INTL") {
9          _mint(msg.sender, 900 * 10**18);
10     }
11 }
```

Risk Level:

Likelihood – 1
Impact - 3

Recommendation:

We recommend using increaseAllowance and decreaseAllowance functions to modify the approval amount instead of using the approve function to modify it.

Status - Acknowledged

The Intelly team has acknowledged the risk, stating that the contract will not be deployed.

F.2 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.7. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 32: Stable.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

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

Status - Fixed

The Intelly team has solved the issue by locking the pragma version to 0.8.7.

G Access.sol

G.1 Floating Pragma [LOW]

Description:

The contract makes use of the floating-point pragma 0.8.7. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts will not unintentionally be deployed using another pragma, which in some cases may be an obsolete version, that may introduce issues to the contract system.

Code:

Listing 33: Access.sol

```
1 //SPDX-License-Identifier: MIT
2 pragma solidity ^0.8.7;
```

Risk Level:

Likelihood – 1
Impact - 2

Recommendation:

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

Status - Fixed

The Intelly team has solved the issue by locking the pragma version to 0.8.7.

4 Tests

Results:

```
Access Contract Unit Tests
  Deploying
    Should deployed with Default Admin Role (80ms)
    Should deployed with Preset Admin Roles (44ms)
    Should deployed with Preset Role Admins for Roles
  Granting Role
    Should grant multiple roles for a single account (190ms)
    Should grant Default Admin Role only with Default Admin Role (497
      ↪ ms)
    Should grant Operator Role with only Default Admin Role (185ms)
    Should grant Moderator Role only with Operator Role (359ms)
    Should grant User Role only with Moderator Role (292ms)
    Should grant Vip Role only with Moderator Role (276ms)
  Revoking Role
    Should revoke Default Admin Role only with Default Admin Role (302
      ↪ ms)
    Should revoke Operator Role only with Default Admin Role (259ms)
    Should revoke Moderator Role only with Operator Role (443ms)
    Should revoke User Role only with Moderator Role (288ms)
    Should revoke Vip Role only with Moderator Role (294ms)
    Should renounce role only from the calling account. (204ms)


Oracle Contract Unit Tests
  Deploying
    Should set permits for tokens on contract deploy
    Should deployed with declared contract addresses
  Read Operations
    Should get fixed prices while 'fix = true' on Oracle Contract (160
      ↪ ms)
    1) Should get prices from Router while 'fix = false' on Oracle
```

```
      ↪ Contract
    Should revert getting price if tokens are not permitted on Oracle
       ↪ Contract (40ms)
    Should revert getting price if both adresses are same on Oracle
       ↪ Contract (39ms)
  Write Operations
    Should set variables only with Operator Role (845ms)
    Should grant permit to tokens only with Operator Role (286ms)
    Should revoke permit from tokens only with Operator Role (222ms)

Exchange Contract Unit Tests
  Deploying
    Should deployed with declared contract addresses and values
  Write Operations
    Should set variables only with Operator Role (575ms)
  Exchange Operations
    Should exchange tokens with fixed prices while 'fix = true' on
       ↪ Oracle Contract (292ms)
   2) Should exchange tokens with routed price while 'fix = false' on
      ↪  Oracle Contract
    Should revert exchange if tokens are not permitted on Oracle
       ↪ Contract (280ms)
    Should revert exchange if both adresses are same

Estate Contract Unit Tests
  Deploying
    Should deployed with declared variables (50ms)
  User Operations
    Should revert NFT transfers if sender is not owner nor approved
       ↪ (691ms)
    Should revert NFT transfers to accounts without User Role on
       ↪ Access Contract (872ms)
    Should revert NFT transfers from accounts without User Role on
       ↪ Access Contract (460ms)
```

```
    Should revert NFT transfers if receiver exceeds Balance Limit on
      ↪ Estate Contract (369ms)
    Should transfer NFT between User Roles (166ms)
    Should pay royalty fee if sender account has no Vip Role on Access
      ↪  Contracrt (304ms)
  Operator Operations
    Should set NFT Uri only with Operator Role (340ms)
    Should pause NFT only with Operator Role (315ms)
    Should unpause NFT only with Operator Role (339ms)
    Should burn NFT only with Operator Role (1601ms)
    Should revert NFT transfer with operator transfer method(OT) calls
      ↪  without Operator Role on Access Contract (1337ms)
    Should transfer NFT with OT only to account that has User or
      ↪ Operator Roles (389ms)
    Should revert NFT transfer with OT to User Role above Balance
      ↪ Limit (241ms)
    Should transfer NFT with OT to Operator Role without limit (80ms)
    Should batch transfer NFT only between Operator Role (2386ms)

Trader Contract Unit Tests
  Deploying
    Should deployed with declared variables
  Listing
    Should revert list NFT without Operator Role on Access Contract
      ↪ (506ms)
    Should revert list NFT without approve to Trader Contract (44ms)
    Should revert list NFT without balance (106ms)
    Should revert list NFT with price under 1 ether (106ms)
    Should list only Approved and Owned NFTs with Operator Role (75ms)
  Purchasing
    Should revert purchase NFT without User Role (200ms)
    Should revert purchase NFT without approved Intelly Token (108ms)
    Should revert purchase NFT without Intelly Token balance (150ms)
    Should revert purchase NFT if User exceeds NFT Balance Limit (191
```

```
      ↪ ms)
    Should purchase listed NFT fractions with an account has User Role
      ↪  and has enough amount of Intelly Token balance and
      ↪ allowance (229ms)


  55 passing (49s)
  2 failing
```

Conclusion:

In order to guarantee the functionality of the contracts in all test cases, we advise fixing the problems that arose when running the tests.

5 Static Analysis (Slither)

Description:

ShellBoxes 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:

```
'npx hardhat compile --force' running
Compiled 24 Solidity files successfully


IAccess is re-used:
 - contracts/Exchange.sol#8-10
 - contracts/Estate.sol#9-11
 - contracts/Oracle.sol#182-184
 - contracts/Estate.sol#9-11
 - contracts/Trader.sol#8-10
 - contracts/Estate.sol#9-11
IToken is re-used:
 - contracts/Exchange.sol#12-18
 - contracts/Estate.sol#20-26
 - contracts/Trader.sol#30-36
 - contracts/Estate.sol#20-26
IOracle is re-used:
 - contracts/Exchange.sol#20-22
 - contracts/Estate.sol#13-18
 - contracts/Trader.sol#26-28
 - contracts/Estate.sol#13-18
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #name-reused

Estate._payFee(address,uint256) (contracts/Estate.sol#110-115) ignores
    ↪ return value by IToken(token).transferFrom(from,creator,price) (
    ↪ contracts/Estate.sol#113)
Exchange.exchange(uint256,address[]) (contracts/Exchange.sol#92-101)
    ↪ ignores return value by IToken(path[0]).transferFrom(_msgSender()
    ↪ ,admin,amount) (contracts/Exchange.sol#95)
Exchange.exchange(uint256,address[]) (contracts/Exchange.sol#92-101)
    ↪ ignores return value by IToken(path[1]).transferFrom(admin,
    ↪ _msgSender(),_getPrice(amount,path)) (contracts/Exchange.sol
    ↪ #96-100)
Trader.purchase(address,uint256,uint256,address[]) (contracts/Trader.sol
    ↪ #142-161) ignores return value by IToken(token).transferFrom(
    ↪ _msgSender(),item.creator,price) (contracts/Trader.sol#153)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unchecked-transfer

Oracle._getFixed(uint256,address[]) (contracts/Oracle.sol#127-135)
    ↪ performs a multiplication on the result of a division:
 -(amount / MEASURE) * ratio (contracts/Oracle.sol#134)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #divide-before-multiply

Reentrancy in Estate.safeTransferFrom(address,address,uint256,uint256,
    ↪ bytes) (contracts/Estate.sol#169-186):
 External calls:
 - _payFee(from,amount) (contracts/Estate.sol#183)
  - IToken(token).transferFrom(from,creator,price) (contracts/Estate.sol
     ↪ #113)
 - _safeTransferFrom(from,to,id,amount,data) (contracts/Estate.sol#185)
  - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data)
     ↪  (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol
```

```
    ↪ #476-484)
 State variables written after the call(s):
 - _safeTransferFrom(from,to,id,amount,data) (contracts/Estate.sol#185)
  - _balances[id][from] = fromBalance - amount (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#178)
  - _balances[id][to] += amount (node_modules/@openzeppelin/contracts/
    ↪ token/ERC1155/ERC1155.sol#180)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #reentrancy-vulnerabilities-1

ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,
   ↪ uint256,bytes).reason (node_modules/@openzeppelin/contracts/token
   ↪ /ERC1155/ERC1155.sol#480) is a local variable never initialized
ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,
   ↪ uint256,bytes).response (node_modules/@openzeppelin/contracts/
   ↪ token/ERC1155/ERC1155.sol#476) is a local variable never
   ↪ initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,
   ↪ uint256[],uint256[],bytes).response (node_modules/@openzeppelin/
   ↪ contracts/token/ERC1155/ERC1155.sol#498) is a local variable
   ↪ never initialized
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,
   ↪ uint256[],uint256[],bytes).reason (node_modules/@openzeppelin/
   ↪ contracts/token/ERC1155/ERC1155.sol#503) is a local variable
   ↪ never initialized
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #uninitialized-local-variables

ERC1155._doSafeTransferAcceptanceCheck(address,address,address,uint256,
   ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/
   ↪ ERC1155/ERC1155.sol#467-486) ignores return value by
   ↪ IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,
   ↪ data) (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155
   ↪ .sol#476-484)
```

```
ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,address,
    ↪ uint256[],uint256[],bytes) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC1155/ERC1155.sol#488-509) ignores return value by
    ↪ IERC1155Receiver(to).onERC1155BatchReceived(operator,from,ids,
    ↪ amounts,data) (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /ERC1155.sol#497-507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #unused-return

Router.constructor(address,address)._token (contracts/local/Router.sol
    ↪ #9) lacks a zero-check on :
  - token = _token (contracts/local/Router.sol#10)
Router.constructor(address,address)._stable (contracts/local/Router.sol
    ↪ #9) lacks a zero-check on :
  - stable = _stable (contracts/local/Router.sol#11)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation

Estate.constructor(address,address,address,address,address,uint256,
    ↪ uint256,uint256)._creator (contracts/Estate.sol#42) lacks a zero-
    ↪ check on :
  - creator = _creator (contracts/Estate.sol#51)
Estate.constructor(address,address,address,address,address,uint256,
    ↪ uint256,uint256)._access (contracts/Estate.sol#43) lacks a zero-
    ↪ check on :
  - access = _access (contracts/Estate.sol#52)
Estate.constructor(address,address,address,address,address,uint256,
    ↪ uint256,uint256)._token (contracts/Estate.sol#44) lacks a zero-
    ↪ check on :
  - token = _token (contracts/Estate.sol#53)
Estate.constructor(address,address,address,address,address,uint256,
    ↪ uint256,uint256)._stable (contracts/Estate.sol#45) lacks a zero-
    ↪ check on :
  - stable = _stable (contracts/Estate.sol#54)
```

```
Estate.constructor(address,address,address,address,address,uint256,
    ↪ uint256,uint256)._oracle (contracts/Estate.sol#46) lacks a zero-
    ↪ check on :
  - oracle = _oracle (contracts/Estate.sol#55)
Exchange.constructor(address,address,address)._admin (contracts/Exchange
    ↪ .sol#35) lacks a zero-check on :
  - admin = _admin (contracts/Exchange.sol#39)
Exchange.constructor(address,address,address)._access (contracts/
    ↪ Exchange.sol#36) lacks a zero-check on :
  - access = _access (contracts/Exchange.sol#40)
Exchange.constructor(address,address,address)._oracle (contracts/
    ↪ Exchange.sol#37) lacks a zero-check on :
  - oracle = _oracle (contracts/Exchange.sol#41)
Exchange.setAccess(address)._access (contracts/Exchange.sol#103) lacks a
    ↪  zero-check on :
  - access = _access (contracts/Exchange.sol#104)
Exchange.setOracle(address)._oracle (contracts/Exchange.sol#107) lacks a
    ↪  zero-check on :
  - oracle = _oracle (contracts/Exchange.sol#108)
Exchange.setAdmin(address)._admin (contracts/Exchange.sol#111) lacks a
    ↪ zero-check on :
  - admin = _admin (contracts/Exchange.sol#112)
Oracle.constructor(address,address,address,address)._access (contracts/
    ↪ Oracle.sol#29) lacks a zero-check on :
  - access = _access (contracts/Oracle.sol#34)
Oracle.constructor(address,address,address,address)._router (contracts/
    ↪ Oracle.sol#30) lacks a zero-check on :
  - router = _router (contracts/Oracle.sol#35)
Oracle.setAccess(address)._access (contracts/Oracle.sol#173) lacks a
    ↪ zero-check on :
  - access = _access (contracts/Oracle.sol#174)
Oracle.setRouter(address)._router (contracts/Oracle.sol#177) lacks a
    ↪ zero-check on :
  - router = _router (contracts/Oracle.sol#178)
```

```
Trader.constructor(address,address,address,address,address)._access (
    ↪ contracts/Trader.sol#56) lacks a zero-check on :
  - access = _access (contracts/Trader.sol#62)
Trader.constructor(address,address,address,address,address)._admin (
    ↪ contracts/Trader.sol#57) lacks a zero-check on :
  - admin = _admin (contracts/Trader.sol#63)
Trader.constructor(address,address,address,address,address)._oracle (
    ↪ contracts/Trader.sol#58) lacks a zero-check on :
  - oracle = _oracle (contracts/Trader.sol#64)
Trader.constructor(address,address,address,address,address)._stable (
    ↪ contracts/Trader.sol#59) lacks a zero-check on :
  - stable = _stable (contracts/Trader.sol#65)
Trader.constructor(address,address,address,address,address)._token (
    ↪ contracts/Trader.sol#60) lacks a zero-check on :
  - token = _token (contracts/Trader.sol#66)
Trader.setAccess(address)._access (contracts/Trader.sol#163) lacks a
    ↪ zero-check on :
  - access = _access (contracts/Trader.sol#164)
Trader.setAdmin(address)._admin (contracts/Trader.sol#167) lacks a zero-
    ↪ check on :
  - admin = _admin (contracts/Trader.sol#168)
Trader.setOracle(address)._oracle (contracts/Trader.sol#171) lacks a
    ↪ zero-check on :
  - oracle = _oracle (contracts/Trader.sol#172)
Trader.setStable(address)._stable (contracts/Trader.sol#175) lacks a
    ↪ zero-check on :
  - stable = _stable (contracts/Trader.sol#176)
Trader.setToken(address)._token (contracts/Trader.sol#179) lacks a zero-
    ↪ check on :
  - token = _token (contracts/Trader.sol#180)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-zero-address-validation
```

```
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address
    ↪ ,uint256,uint256,bytes).response (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#476)' in ERC1155.
    ↪ _doSafeTransferAcceptanceCheck(address,address,address,uint256,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/
    ↪ ERC1155/ERC1155.sol#467-486) potentially used before declaration:
    ↪  response != IERC1155Receiver.onERC1155Received.selector (
    ↪ node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol
    ↪ #477)
Variable 'ERC1155._doSafeTransferAcceptanceCheck(address,address,address
    ↪ ,uint256,uint256,bytes).reason (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#480)' in ERC1155.
    ↪ _doSafeTransferAcceptanceCheck(address,address,address,uint256,
    ↪ uint256,bytes) (node_modules/@openzeppelin/contracts/token/
    ↪ ERC1155/ERC1155.sol#467-486) potentially used before declaration:
    ↪  revert(string)(reason) (node_modules/@openzeppelin/contracts/
    ↪ token/ERC1155/ERC1155.sol#481)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
    ↪ address,uint256[],uint256[],bytes).response (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#498)' in
    ↪ ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
    ↪ address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#488-509) potentially used
    ↪ before declaration: response != IERC1155Receiver.
    ↪ onERC1155BatchReceived.selector (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#500)
Variable 'ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
    ↪ address,uint256[],uint256[],bytes).reason (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#503)' in
    ↪ ERC1155._doSafeBatchTransferAcceptanceCheck(address,address,
    ↪ address,uint256[],uint256[],bytes) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#488-509) potentially used
    ↪ before declaration: revert(string)(reason) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#504)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #pre-declaration-usage-of-local-variables

Reentrancy in Estate.safeTransferFrom(address,address,uint256,uint256,
    ↪ bytes) (contracts/Estate.sol#169-186):
 External calls:
 - _payFee(from,amount) (contracts/Estate.sol#183)
  - IToken(token).transferFrom(from,creator,price) (contracts/Estate.sol
      ↪ #113)
 - _safeTransferFrom(from,to,id,amount,data) (contracts/Estate.sol#185)
  - IERC1155Receiver(to).onERC1155Received(operator,from,id,amount,data)
      ↪  (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol
      ↪ #476-484)
 Event emitted after the call(s):
 - TransferSingle(operator,from,to,id,amount) (node_modules/
     ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#182)
  - _safeTransferFrom(from,to,id,amount,data) (contracts/Estate.sol#185)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #reentrancy-vulnerabilities-3

Address.verifyCallResult(bool,bytes,string) (node_modules/@openzeppelin/
    ↪ contracts/utils/Address.sol#201-221) uses assembly
 - INLINE ASM (node_modules/@openzeppelin/contracts/utils/Address.sol
     ↪ #213-216)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #assembly-usage

Different versions of Solidity is used:
 - Version used: ['^0.8.0', '^0.8.1', '^0.8.7']
 - ^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/security/ReentrancyGuard
    ↪ .sol#4)
 - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.
    ↪ sol#4)
 - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/IERC1155.
    ↪ sol#4)
 - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/
    ↪ IERC1155Receiver.sol#4)
 - ^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155/extensions
    ↪ /IERC1155MetadataURI.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)
 - ^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address.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)
 - ^0.8.7 (contracts/Access.sol#2)
 - ^0.8.7 (contracts/Estate.sol#2)
 - ^0.8.7 (contracts/Exchange.sol#2)
 - ^0.8.7 (contracts/Oracle.sol#2)
 - ^0.8.7 (contracts/Trader.sol#2)
 - ^0.8.7 (contracts/local/Stable.sol#2)
 - ^0.8.7 (contracts/local/Token.sol#2)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
   ↪ #different-pragma-directives-are-used

Estate._beforeTokenTransfer(address,address,address,uint256[],uint256[],
   ↪ bytes) (contracts/Estate.sol#95-104) is never used and should be
```

```
    ↪ removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #dead-code

Pragma version^0.8.7 (contracts/local/Router.sol#2) necessitates a
    ↪ version too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6
solc-0.8.7 is not recommended for deployment
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
    ↪ AccessControl.sol#4) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/access/
    ↪ IAccessControl.sol#4) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/
    ↪ Pausable.sol#4) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/security/
    ↪ ReentrancyGuard.sol#4) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /ERC1155.sol#4) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /IERC1155.sol#4) necessitates a version too recent to be trusted.
    ↪  Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /IERC1155Receiver.sol#4) necessitates a version too recent to be
    ↪ trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC1155
    ↪ /extensions/IERC1155MetadataURI.sol#4) necessitates a version too
```

```
    ↪  recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ ERC20.sol#4) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ IERC20.sol#4) necessitates a version too recent to be trusted.
    ↪ Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ extensions/IERC20Metadata.sol#4) necessitates a version too
    ↪ recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.1 (node_modules/@openzeppelin/contracts/utils/Address
    ↪ .sol#4) necessitates a version too recent to be trusted. Consider
    ↪  deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Context
    ↪ .sol#4) necessitates a version too recent to be trusted. Consider
    ↪  deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/Strings
    ↪ .sol#4) necessitates a version too recent to be trusted. Consider
    ↪  deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
    ↪ introspection/ERC165.sol#4) necessitates a version too recent to
    ↪ be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.0 (node_modules/@openzeppelin/contracts/utils/
    ↪ introspection/IERC165.sol#4) necessitates a version too recent to
    ↪  be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.7 (contracts/Access.sol#2) necessitates a version too
    ↪  recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.7 (contracts/Estate.sol#2) necessitates a version too
    ↪  recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.7 (contracts/Exchange.sol#2) necessitates a version
    ↪ too recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.7 (contracts/Oracle.sol#2) necessitates a version too
    ↪  recent to be trusted. Consider deploying with 0.6.12/0.7.6
```

```
Pragma version^0.8.7 (contracts/Trader.sol#2) necessitates a version too
    ↪  recent to be trusted. Consider deploying with 0.6.12/0.7.6
Pragma version^0.8.7 (contracts/local/Stable.sol#2) necessitates a
    ↪ version too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6
Pragma version^0.8.7 (contracts/local/Token.sol#2) necessitates a
    ↪ version too recent to be trusted. Consider deploying with
    ↪ 0.6.12/0.7.6
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #incorrect-versions-of-solidity

Low level call in Address.sendValue(address,uint256) (node_modules/
    ↪ @openzeppelin/contracts/utils/Address.sol#60-65):
 - (success) = recipient.call{value: amount}() (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#63)
Low level call in Address.functionCallWithValue(address,bytes,uint256,
    ↪ string) (node_modules/@openzeppelin/contracts/utils/Address.sol
    ↪ #128-139):
 - (success,returndata) = target.call{value: value}(data) (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#137)
Low level call in Address.functionStaticCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#157-166):
 - (success,returndata) = target.staticcall(data) (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#164)
Low level call in Address.functionDelegateCall(address,bytes,string) (
    ↪ node_modules/@openzeppelin/contracts/utils/Address.sol#184-193):
 - (success,returndata) = target.delegatecall(data) (node_modules/
     ↪ @openzeppelin/contracts/utils/Address.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #low-level-calls

Oracle (contracts/Oracle.sol#8-180) should inherit from IOracle (
    ↪ contracts/Estate.sol#13-18)
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #missing-inheritance

Parameter Estate.setFee(uint256)._fee (contracts/Estate.sol#137) is not
    ↪ in mixedCase
Parameter Estate.setLimit(uint256)._limit (contracts/Estate.sol#141) is
    ↪ not in mixedCase
Parameter Estate.setPath(address[])._path (contracts/Estate.sol#149) is
    ↪ not in mixedCase
Parameter Exchange.setAccess(address)._access (contracts/Exchange.sol
    ↪ #103) is not in mixedCase
Parameter Exchange.setOracle(address)._oracle (contracts/Exchange.sol
    ↪ #107) is not in mixedCase
Parameter Exchange.setAdmin(address)._admin (contracts/Exchange.sol#111)
    ↪  is not in mixedCase
Parameter Exchange.setMin(uint256)._min (contracts/Exchange.sol#115) is
    ↪ not in mixedCase
Parameter Exchange.setMax(uint256)._max (contracts/Exchange.sol#119) is
    ↪ not in mixedCase
Parameter Oracle.setFromRatio(uint256)._fromRatio (contracts/Oracle.sol
    ↪ #165) is not in mixedCase
Parameter Oracle.setToRatio(uint256)._toRatio (contracts/Oracle.sol#169)
    ↪  is not in mixedCase
Parameter Oracle.setAccess(address)._access (contracts/Oracle.sol#173)
    ↪ is not in mixedCase
Parameter Oracle.setRouter(address)._router (contracts/Oracle.sol#177)
    ↪ is not in mixedCase
Parameter Trader.setAccess(address)._access (contracts/Trader.sol#163)
    ↪ is not in mixedCase
Parameter Trader.setAdmin(address)._admin (contracts/Trader.sol#167) is
    ↪ not in mixedCase
Parameter Trader.setOracle(address)._oracle (contracts/Trader.sol#171)
    ↪ is not in mixedCase
```

```
Parameter Trader.setStable(address)._stable (contracts/Trader.sol#175)
    ↪ is not in mixedCase
Parameter Trader.setToken(address)._token (contracts/Trader.sol#179) is
    ↪ not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #conformance-to-solidity-naming-conventions

Oracle.slitherConstructorConstantVariables() (contracts/Oracle.sol
    ↪ #8-180) uses literals with too many digits:
 - MEASURE = 1000000 (contracts/Oracle.sol#13)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #too-many-digits

getAmountsOut(uint256,address[]) should be declared external:
 - Router.getAmountsOut(uint256,address[]) (contracts/local/Router.sol
    ↪ #14-27)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external

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)
uri(uint256) should be declared external:
 - ERC1155.uri(uint256) (node_modules/@openzeppelin/contracts/token/
    ↪ ERC1155/ERC1155.sol#59-61)
balanceOfBatch(address[],uint256[]) should be declared external:
 - ERC1155.balanceOfBatch(address[],uint256[]) (node_modules/
    ↪ @openzeppelin/contracts/token/ERC1155/ERC1155.sol#82-98)
```

```
setApprovalForAll(address,bool) should be declared external:
 - ERC1155.setApprovalForAll(address,bool) (node_modules/@openzeppelin/
    ↪ contracts/token/ERC1155/ERC1155.sol#103-105)
safeTransferFrom(address,address,uint256,uint256,bytes) should be
   ↪ declared external:
 - ERC1155.safeTransferFrom(address,address,uint256,uint256,bytes) (
    ↪ node_modules/@openzeppelin/contracts/token/ERC1155/ERC1155.sol
    ↪ #117-129)
 - Estate.safeTransferFrom(address,address,uint256,uint256,bytes) (
    ↪ contracts/Estate.sol#169-186)
safeBatchTransferFrom(address,address,uint256[],uint256[],bytes) should
   ↪ be declared external:
 - ERC1155.safeBatchTransferFrom(address,address,uint256[],uint256[],
    ↪ bytes) (node_modules/@openzeppelin/contracts/token/ERC1155/
    ↪ ERC1155.sol#134-146)
 - Estate.safeBatchTransferFrom(address,address,uint256[],uint256[],
    ↪ bytes) (contracts/Estate.sol#207-217)
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)
decimals() should be declared external:
 - ERC20.decimals() (node_modules/@openzeppelin/contracts/token/ERC20/
    ↪ ERC20.sol#87-89)
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)
setFee(uint256) should be declared external:
 - Estate.setFee(uint256) (contracts/Estate.sol#137-139)
setLimit(uint256) should be declared external:
 - Estate.setLimit(uint256) (contracts/Estate.sol#141-143)
setURI(string) should be declared external:
 - Estate.setURI(string) (contracts/Estate.sol#145-147)
setPath(address[]) should be declared external:
 - Estate.setPath(address[]) (contracts/Estate.sol#149-151)
pause() should be declared external:
 - Estate.pause() (contracts/Estate.sol#153-155)
unpause() should be declared external:
 - Estate.unpause() (contracts/Estate.sol#157-159)
burn(address,uint256,uint256) should be declared external:
 - Estate.burn(address,uint256,uint256) (contracts/Estate.sol#161-167)
operatorTransfer(address,address,uint256,uint256,bytes) should be
   ↪ declared external:
 - Estate.operatorTransfer(address,address,uint256,uint256,bytes) (
    ↪ contracts/Estate.sol#188-205)
exchange(uint256,address[]) should be declared external:
 - Exchange.exchange(uint256,address[]) (contracts/Exchange.sol#92-101)
```

```
setAccess(address) should be declared external:
 - Exchange.setAccess(address) (contracts/Exchange.sol#103-105)
setOracle(address) should be declared external:
 - Exchange.setOracle(address) (contracts/Exchange.sol#107-109)
setAdmin(address) should be declared external:
 - Exchange.setAdmin(address) (contracts/Exchange.sol#111-113)
setMin(uint256) should be declared external:
 - Exchange.setMin(uint256) (contracts/Exchange.sol#115-117)
setMax(uint256) should be declared external:
 - Exchange.setMax(uint256) (contracts/Exchange.sol#119-121)
grantPermit(bytes32,address) should be declared external:
 - Oracle.grantPermit(bytes32,address) (contracts/Oracle.sol#95-101)
revokePermit(bytes32,address) should be declared external:
 - Oracle.revokePermit(bytes32,address) (contracts/Oracle.sol#103-109)
getPrice(uint256,address[]) should be declared external:
 - Oracle.getPrice(uint256,address[]) (contracts/Oracle.sol#147-159)
switchFix(bool) should be declared external:
 - Oracle.switchFix(bool) (contracts/Oracle.sol#161-163)
setFromRatio(uint256) should be declared external:
 - Oracle.setFromRatio(uint256) (contracts/Oracle.sol#165-167)
setToRatio(uint256) should be declared external:
 - Oracle.setToRatio(uint256) (contracts/Oracle.sol#169-171)
setAccess(address) should be declared external:
 - Oracle.setAccess(address) (contracts/Oracle.sol#173-175)
setRouter(address) should be declared external:
 - Oracle.setRouter(address) (contracts/Oracle.sol#177-179)
list(address,address,uint256,uint256) should be declared external:
 - Trader.list(address,address,uint256,uint256) (contracts/Trader.sol
    ↪ #119-140)
purchase(address,uint256,uint256,address[]) should be declared external:
 - Trader.purchase(address,uint256,uint256,address[]) (contracts/Trader.
    ↪ sol#142-161)
setAccess(address) should be declared external:
 - Trader.setAccess(address) (contracts/Trader.sol#163-165)
```

```
setAdmin(address) should be declared external:
 - Trader.setAdmin(address) (contracts/Trader.sol#167-169)
setOracle(address) should be declared external:
 - Trader.setOracle(address) (contracts/Trader.sol#171-173)
setStable(address) should be declared external:
 - Trader.setStable(address) (contracts/Trader.sol#175-177)
setToken(address) should be declared external:
 - Trader.setToken(address) (contracts/Trader.sol#179-181)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation
    ↪ #public-function-that-could-be-declared-external
. analyzed (29 contracts with 75 detectors), 146 result(s) found
```

Conclusion:

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

6 Conclusion

In this audit, we examined the design and implementation of INTELLY contract and discovered several issues of varying severity. Intelly team addressed most of the issues raised in the initial report and implemented the necessary fixes.

The present code base is well-structured and ready for the mainnet.

For a Contract Audit, contact us at contact@shellboxes.com