Competitive Security Assessment

Gameland

May 6th, 2023

Secure3

secure3.io

Summary

This report is prepared for the project to identify vulnerabilities and issues in the smart contract source code. A group of NDA covered experienced security experts have participated in the Secure3's Audit Contest to find vulnerabilities and optimizations. Secure3 team has participated in the contest process as well to provide extra auditing coverage and scrutiny of the finding submissions.

The comprehensive examination and auditing scope includes:

• Cross checking contract implementation against functionalities described in the documents and white paper disclosed by the project owner.

• Contract Privilege Role Review to provide more clarity on smart contract roles and privilege.

• Using static analysis tools to analyze smart contracts against common known vulnerabilities patterns.

• Verify the code base is compliant with the most up-to-date industry standards and security best practices.

• Comprehensive line-by-line manual code review of the entire codebase by industry experts.

The security assessment resulted in findings that are categorized in four severity levels: Critical, Medium, Low, Informational. For each of the findings, the report has included recommendations of fix or mitigation for security and best practices.

Overview

Project Detail

Project Name	Gameland
Platform & Language	Solidity
Codebase	• https://github.com/Gameland0/smart-contract • 2c524d8da26876951350afb8c5310bc232ea4a51 • e97bf4649181f8afc6c31b9b10f5fac81f8ca674
Audit Methodology	• Audit Contest • Business Logic and Code Review • Privileged Roles Review • Static Analysis

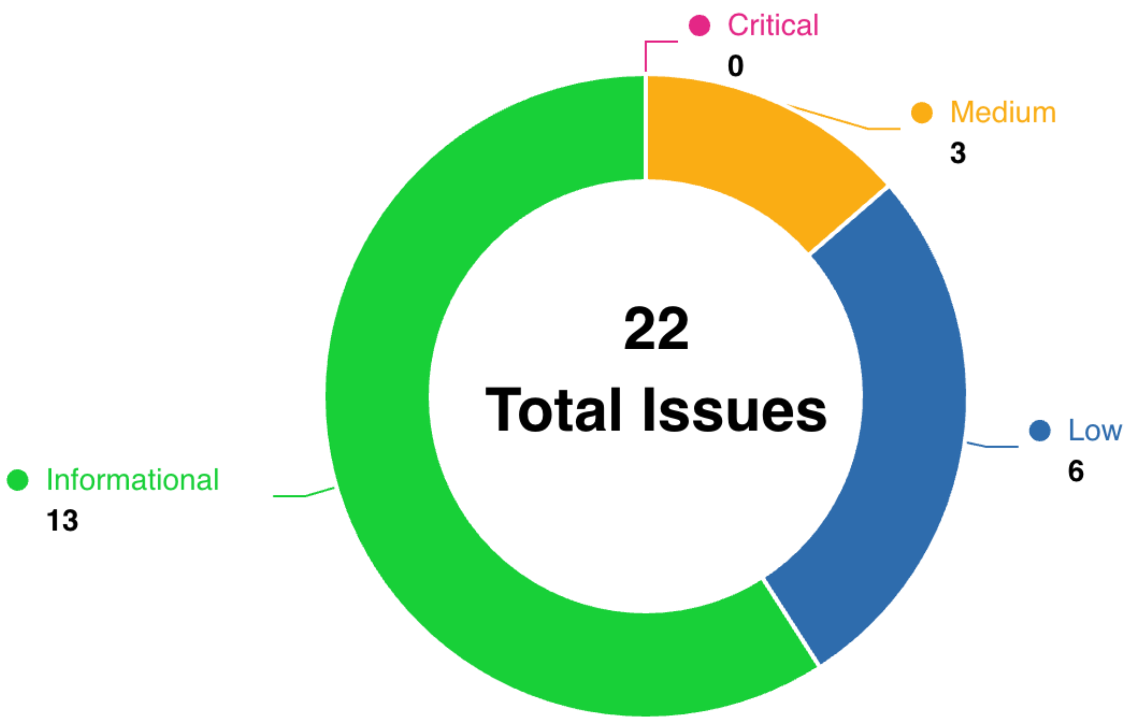
Code Vulnerability Review Summary

Vulnerability Level	Total	Reported	Acknowledged	Fixed	Mitigated	Declined
Critical	0	0	0	0	0	0
Medium	3	0	0	2	0	1
Low	6	0	1	2	0	3
Informational	13	0	1	6	0	6

Audit Scope

File	Commit Hash
code/payment_contract.sol	**2c524d8da26876951350afb8c5310bc232ea4a51**

Code Assessment Findings



ID	Name	Category	Severity	Status	Contributor
GML-1	Constant naming not correctly	Code Style	Informational	Fixed	SAir
GML-2	Ether may locked in `GameLand_verify` contract	Logical	Medium	Declined	w2ning, 0xzoobi
GML-3	Gas Optimization in `payment_contract::constructor`	Gas Optimization	Informational	Fixed	0xzoobi
GML-4	Gas Optimization: use `external` instead of `public`	Gas Optimization	Informational	Declined	0xzoobi
GML-5	Gas Optimization: use calldata as much as possible	Gas Optimization	Informational	Fixed	Xi_Zi

GML-6	Gas Optimization: using `immutable` instead of null	Gas Optimization	Informational	Fixed	Xi_Zi
GML-7	Gas Optimization:Cache array length outside for loop in `GameLand_verify:batch_set_address_amount`	Gas Optimization	Informational	Acknowledged	Xi_Zi
GML-8	Logic Error in `GameLand_verify::verify_address_amount`	Logical	Medium	Fixed	w2ning
GML-9	Logic error in `GameLand_verify::get_whethertobuy`: return `index i` instead of 1	Logical	Low	Fixed	0xzoobi
GML-10	Missing `require` check in `GameLand_verify::verify_address_amount` after `call`	Logical	Medium	Fixed	w2ning
GML-11	Missing 0 address check in `GameLand_verify:updateOwner`	Logical	Low	Acknowledged	Xi_Zi
GML-12	Missing Event in `GameLand_verify` contract	Code Style	Informational	Declined	w2ning, Xi_Zi, 0xzoobi
GML-13	Missing check array parameter length in `GameLand_verify::batch_set_address_amount`	Logical	Informational	Fixed	w2ning, Xi_Zi
GML-14	Missing update `verify_info::dt` and `address_amount_info::sl` in `GameLand_verify::set_address_amount` if zt != 999999999	Logical	Low	Declined	0xzoobi
GML-15	Not following the pull-over-push pattern in `GameLand_verify::verify_address_amount`	Gas Optimization	Informational	Declined	0xzoobi
GML-16	Precision issue in `GameLand_verify::verify_address_amount`	Logical	Low	Fixed	w2ning, 0xzoobi
GML-17	Redundant use of receive and fallback in `GameLand_verify`	Gas Optimization	Informational	Declined	0xzoobi
GML-18	Reentrancy risk in `GameLand_verify` contract `verify_address_amount` function	Reentrancy	Informational	Fixed	SAir

GML-19	**Unuse the latest solidity version**	**Logical**	**Informational**	**Declined**	**0xzoobi**
GML-20	**Unused function: `GameLand_verify::erc20approve`**	**Gas Optimization**	**Informational**	**Declined**	**w2ning**
GML-21	**Using a state variable to track the balance instead of `address(this).balance`**	**Logical**	**Low**	**Declined**	**0xzoobi**
GML-22	**for loop unlimited number of iterations risk in `GameLand_verify` contract `find_address` and `batch_set_address_amount` function**	**Dos**	**Low**	**Declined**	**SAir**

GML-1: Constant naming not correctly

Category	Severity	Code Reference	Status	Contributor
Code Style	**Informational**	• **code/payment_contract.sol#L25** • **code/payment_contract.sol#L26** • **code/payment_contract.sol#L29**	**Fixed**	**SAir**

Code

```
25:    address[] address_sz;

26:    struct address_amount_info

29:        verify_info[] vi;
```

Description

SAir : In Solidity, the naming convention for variable names is to use a combination of lowercase letters and underscores, but in this contract, there are several variable naming conventions, such as `address_sz`, it is recommended to change it to addressList or addressSZ.

Recommendation

SAir : Uniformly use Solidity naming conventions to avoid variable names that are too short and confusing.Constant names should be in camelCase.

Client Response

Complete the modification

GML-2:Ether may locked in `GameLand_verify` contract

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/payment_contract.sol#L7 • code/payment_contract.sol#L9 • code/payment_contract.sol#L213	Declined	w2ning, 0xzoobi

Code

```
7:    fallback() external payable {}

9:    receive() external payable {}

213:        return address(this).balance;
```

Description

w2ning : Contract `GameLand_verify` has payable functions and a read-only function used to read the balance of the contract.

But does not have a function to withdraw the ether

Every Ether sent to `GameLand_verify` contract will be lost.

```
contract GameLand_verify {

    fallback() external payable {}

    receive() external payable {}

     ...

    function collatoralbalance() public view returns (uint256) {
        return address(this).balance;
    }

}
```

0xzoobi : Contract `payment_contract.sol` has `payable` functions but does not have a function to withdraw the ether. As a result any ether sent to the contract is locked forever and cannot be withdrawn.

Recommendation

w2ning : Remove the payable attribute or add a withdraw function.

Consider below fix in the `GameLand_verify` contract

```
// fix: Add a withdraw function.
function withdraw(address to, uint256 value) onlyOwner{

    to.transfer(value);

}
```

0xzoobi : add a withdraw function protected by `onlyOwner` or `onlyGove` to transfer the ether.

Code Fix:

```
    function withdraw(uint amount) onlyOwner returns(bool) {
        require(amount <= this.balance);
        owner.transfer(amount);
        return true;
    }
```

Client Response

Contracts do not require this feature

GML-3:Gas Optimization in `payment_contract::constructor`

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/payment_contract.sol#L260 • code/payment_contract.sol#L270	Fixed	0xzoobi

Code

```
260:        ERC20 u = ERC20(usdt);

270:        ERC20 u = ERC20(usdt);
```

Description

0xzoobi : The `erc20allowance` and `erc20getBalance` are currently reusing the same code to initialize the usdt token. Defining it once in the constructor can save some gas fees and provide better code readability.

Recommendation

0xzoobi : Define the usdt token in constructor.

Sample Fix:

```
ERC20 usdt_token;

 constructor(address _u, address _rev) {
        usdt_token= ERC20 (_u);
    }

//Modifed erc20getBalance

    function erc20getBalance(address dz) public view returns (uint256) {
        return usdt_token.balanceOf(dz);
    }
```

Client Response

Complete the modification

GML-4:Gas Optimization: use `external` instead of `public`

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/payment_contract.sol#L104-L106	Declined	0xzoobi

Code

```
104:    function set_baseprice(uint newbaseprice) public onlyGove{
105:        baseprice = newbaseprice;
106:    }
```

Description

0xzoobi : `public` identifier can be used when both smart contract and EOA users are expected to call the functions. If the functions in the `payment_contract.sol` are expected to be called by only EOA users they can be declared as `external`. This will save some amount of gas.

Recommendation

0xzoobi : Use `external` identifier over `public`.

Sample Fix:

```
function set_baseprice(uint newbaseprice) external onlyGove{
        baseprice = newbaseprice;
}
```

Client Response

The contract needs to be this way

GML-5:Gas Optimization: use calldata as much as possible

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	**Informational**	• **code/payment_contract.sol#L75-L102**	**Fixed**	**Xi_Zi**

Code

```
75:    function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onlyGove{
76:        for(uint i =0; i
```

Description

Xi_Zi : Using calldata saves gas more than using memory.

Recommendation

Xi_Zi : Using calldata

Client Response

Complete the modification

GML-6:Gas Optimization: using `immutable` instead of null

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/payment_contract.sol#L14	Fixed	Xi_Zi

Code

```
14:    address usdt;
```

Description

Xi_Zi : Variables set in constructor that are never modified in the contract should be immutable.

Recommendation

Xi_Zi : Use immutable to modify.

Consider below fix in the `payment_contract`

```
    address immutable usdt;
```

Client Response

Complete the modification

GML-7:Gas Optimization:Cache array length outside for loop in `GameLand_verify:batch_set_address_amount`

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/payment_contract.sol#L66-L71 • code/payment_contract.sol#L76-L100	Acknowledged	Xi_Zi

Code

```
66:        for(uint256 i=0;i
```

Description

Xi_Zi : The loop structure can be optimized.

Recommendation

Xi_Zi : Assigning array length to memory, and using unchecked.
Consider below fix in the `payment_contract.batch_set_address_amount()` function

```
    function batch_set_address_amount(uint256[] calldata amounts, address[] calldata adds) public on
lyGove{
        uint256 len= adds.length;
        for(uint i =0; i<len;)
        {
            address add = adds[i];
            uint amount = amounts[i];
            uint256 zt = find_address(add);
            if(zt == 999999999)
            {
                uint256 asl = address_sz.length;
                address_sz.push(add);

                address_info[asl].add = add;
                address_info[asl].price = amount;
                verify_info memory vvi;
                vvi.gm_address = add;
                vvi.dt = uint256(block.timestamp);
                vvi.price = amount;

                address_info[asl].vi.push(vvi);
                address_info[asl].sl += 1;
            }
            else{
                address_info[zt].price = amount;

            }
            unchecked{++i;}
        }

    }
```

Client Response

Acknowledged

GML-8:Logic Error in `GameLand_verify::verify_address_amount`

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/payment_contract.sol#L118 • code/payment_contract.sol#L142	Fixed	w2ning

Code

```
118:            bool success = erc20transferFrom(msg.sender,address(this),re);

142:            bool success = erc20transferFrom(msg.sender,address(this),re);
```

Description

w2ning : On line 118: The value of variable `re` should be `address_info[zt].price` instead of allowance of USDT On line 142: The value of variable `re` should be `baseprice` instead of allowance of USDT

```
function verify_address_amount(address mdd_address) public{
    uint256 re = erc20allowance(msg.sender,address(this));
    uint256 zt = find_address(mdd_address);
    if(zt != 999999999)
    {
        require(
        re >= address_info[zt].price,
        "Not enough amount"
        );

        // The value of variable `re` should be address_info[zt].price instead of allowance of USDT
        bool success = erc20transferFrom(msg.sender,address(this),re);

        ...

  else{
        require(
        re >= baseprice,
        "Not enough amount"
        );

        // The value of variable `re` should be baseprice instead of allowance of USDT
        bool success = erc20transferFrom(msg.sender,address(this),re);

}
```

Recommendation

w2ning : Consider below fix in the `GameLand_verify.verify_address_amount()` function

```
function verify_address_amount(address mdd_address) public{
    uint256 re = erc20allowance(msg.sender,address(this));
    uint256 zt = find_address(mdd_address);
    if(zt != 999999999)
    {
        require(
        re >= address_info[zt].price,
        "Not enough amount"
        );

        // fix: Assign the value of address_info[zt].price to the variable 're'.
        re = address_info[zt].price;

        bool success = erc20transferFrom(msg.sender,address(this),re);

        ...

  else{
        require(
        re >= baseprice,
        "Not enough amount"
        );

        // fix: Assign the value of baseprice to the variable 're'.
        re = baseprice;
        bool success = erc20transferFrom(msg.sender,address(this),re);

}
```

Client Response

Complete the modification

GML-9:Logic error in `GameLand_verify::get_whethertobuy`: return `index i` instead of 1

Category	Severity	Code Reference	Status	Contributor
Logical	Low	• code/payment_contract.sol#L181	Fixed	0xzoobi

Code

```
181:                        return 1;
```

Description

0xzoobi : `get_whethertobuy` takes an address as a parameter and then is expected to return the index at which `mdadd` is present. but currently it just returns the value 1.

Recommendation

0xzoobi : Modify

```
function get_whethertobuy(address mdadd) view public returns(uint){
        address add = msg.sender;
        uint256 zt = find_address(mdadd);
        if(zt != 999999999)
        {
            for(uint i=0;i<address_info[zt].vi.length;i++)
            {
                if(address_info[zt].vi[i].gm_address == add)
                {
                    return 1;
                }
            }
        }
        return 999999999;
    }
```

`i` instead of `1`

Client Response

Complete the modification

GML-10:Missing `require` check in `GameLand_verify::verify_address_amount` after `call`

Category	Severity	Code Reference	Status	Contributor
Logical	Medium	• code/payment_contract.sol#L119 • code/payment_contract.sol#L143	Fixed	w2ning

Code

```
119:            (success, "transfer error!");

143:            (success, "transfer error!");
```

Description

w2ning : Missing keyword `require`

```
function verify_address_amount(address mdd_address) public{
    uint256 re = erc20allowance(msg.sender,address(this));
    uint256 zt = find_address(mdd_address);
    if(zt != 999999999)
    {
        require(
        re >= address_info[zt].price,
        "Not enough amount"
        );

        bool success = erc20transferFrom(msg.sender,address(this),re);

        //  Missing keyword require
        (success, "transfer error!");
```

Recommendation

w2ning : Add keyword `require`

Consider below fix in the `GameLand_verify.verify_address_amount()` function

```
function verify_address_amount(address mdd_address) public{
    uint256 re = erc20allowance(msg.sender,address(this));
    uint256 zt = find_address(mdd_address);
    if(zt != 999999999)
    {
        require(
        re >= address_info[zt].price,
        "Not enough amount"
        );

        bool success = erc20transferFrom(msg.sender,address(this),re);

        //  fix: Add keyword require
        require(success, "transfer error!");
```

Client Response

Complete the modification

GML-11:Missing 0 address check in `GameLand_verify:updateOwner`

Category	Severity	Code Reference	Status	Contributor
Logical	**Low**	• **code/payment_contract.sol#L204-L206**	**Acknowledged**	**Xi_Zi**

Code

```
204:    function updateOwner(address _Owner) public onlyOwner{
205:        owner = _Owner;
206:    }
```

Description

Xi_Zi : The owner permission may be lost because the 0 address check is missing.

Recommendation

Xi_Zi : Add the 0 address check

Client Response

Acknowledged

GML-12:Missing Event in `GameLand_verify` contract

Category	Severity	Code Reference	Status	Contributor
Code Style	Informational	• code/payment_contract.sol#L41 • code/payment_contract.sol#L75 • code/payment_contract.sol#L75 • code/payment_contract.sol#L105 • code/payment_contract.sol#L106 • code/payment_contract.sol#L108 • code/payment_contract.sol#L109 • code/payment_contract.sol#L205 • code/payment_contract.sol#L206 • code/payment_contract.sol#L209 • code/payment_contract.sol#L210	Declined	w2ning, Xi_Zi, 0xzoobi

Code

```
41:    function set_address_amount(uint256 amount) public{

75 :    function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onlyGove{

75:    function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onlyGove{

105:        baseprice = newbaseprice;

106:    }

108:    function verify_address_amount(address mdd_address) public{

109:        uint256 re = erc20allowance(msg.sender,address(this));

205:        owner = _Owner;

206:    }

209:        governance = _gove;

210:    }
```

Description

w2ning : Since those functions change the storage, it is best practice to emit an event for each functions which changes the storage.

```
function set_address_amount(uint256 amount) public{}

function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onlyGove{}

function set_baseprice(uint newbaseprice) public onlyGove{}

function verify_address_amount(address mdd_address) public{}

function updateOwner(address _Owner) public onlyOwner{}

function updategove(address _gove) public onlyOwner{}
```

Xi_Zi : set_address_amount(),batch_set_address_amount(),verify_address_amount() Function execution success requires an event to record.
0xzoobi : Every project must follow the template wherein they emit events on important changes and updates happening in the dapp. Emitting events allows monitoring activities with off-chain monitoring tools. It also provides transparency to the users when some important changes are made to the protocol.

Recommendation

w2ning : Emit an event for each functions which changes the storage.
Xi_Zi : Add an event.
0xzoobi : Emit an event to track the events.

Client Response

Contracts do not require this feature

GML-13:Missing check array parameter length in `GameLand_verify::batch_set_address_amount`

Category	Severity	Code Reference	Status	Contributor
Logical	**Informational**	• **code/payment_contract.sol#L75** • **code/payment_contract.sol#L75-L102**	**Fixed**	**w2ning, Xi_Zi**

Code

```
75:    function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onl
yGove{

75:    function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onl
yGove{
76:        for(uint i =0; i
```

Description

w2ning : The `batch_set_address_amount` function lacks verification of whether the lengths of the two arrays in the incoming parameters are equal.

```
function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onlyGove{
    for(uint i =0; i<adds.length;i++)
    {
        address add = adds[i];
        uint amount = amounts[i];
        uint256 zt = find_address(add);
        if(zt == 999999999)
        {
```

Xi_Zi : The lengths of the amounts passed to batch_set_address_amount() and adds are equal, which may cause an incorrect setting.

Recommendation

w2ning : Add verification

Consider below fix in the `GameLand_verify.batch_set_address_amount()` function

```
function batch_set_address_amount(uint256[] memory amounts, address[] memory adds) public onlyGove{

    // fix: Add verification
    require(amounts.length == adds.length, "The lengths of the two arrays must be equal");

    for(uint i =0; i<adds.length;i++)
    {
        address add = adds[i];
        uint amount = amounts[i];
        uint256 zt = find_address(add);
        if(zt == 999999999)
        {
```

Xi_Zi : Add amounts and adds equal length.

Client Response

Complete the modification

GML-14:Missing update `verify_info::dt` and `address_amount_info::sl` in `GameLand_verify::set_address_amount` if zt != 999999999

Category	Severity	Code Reference	Status	Contributor
Logical	Low	• code/payment_contract.sol#L60 • code/payment_contract.sol#L97	Declined	0xzoobi

Code

```
60:            address_info[zt].price = amount;

97:                address_info[zt].price = amount;
```

Description

0xzoobi : When `set_address_amount` and `batch_set_address_amount` is called, the `vvi.dt = uint256(block.timestamp);` and `address_info[asl].sl += 1;` is set in case where `find_address` returns 999999999. If not, the code currently just updates the address_info's price parameter and `dt` of verify_info and `sl` of address_amount_info is never updated.

Recommendation

0xzoobi : Update the verify_info's dt timestamp and address_amount_info's sl in case where `find_address` does not return 999999999.

Client Response

The contract needs to be this way

GML-15:Not following the pull-over-push pattern in `GameLand_verify::verify_address_amount`

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/payment_contract.sol#L122 • code/payment_contract.sol#L124 • code/payment_contract.sol#L146 • code/payment_contract.sol#L148	Declined	0xzoobi

Code

```
122:            bool success2 =  erc20transfer(mdd_address, re);

124:            bool success3 =  erc20transfer(rev, fee);

146:            bool success2 =  erc20transfer(mdd_address, re);

148:            bool success3 =  erc20transfer(rev, fee);
```

Description

0xzoobi : There are three transfers taking place. First is the `erc20transferFrom` from `msg.sender` to the `payment_contract.sol`. Second is `erc20transfer` from `payment_contract.sol` to `mdd_address` Third is `erc20transfer` from `payment_contract.sol` to `rev`.
After the first transfer of tokens, A better approach would be to store the balances to transfers and only transfer it when the supposed user calls it. This basically improves the UX for the user and saves a ton of gas to the caller of the function.

Recommendation

0xzoobi : Follow pull-over-push pattern in `verify_address_amount` function
Sample Fix:

```
mapping(address => uint256) balances; //mapping to track user balances

//add a new withdraw function to allow users to withdraw the tokens
function withdraw_amount() external {
      require(balances[msg.sender] > 0, "Zero balance");
      uint256 amount_to_transfer = balances[msg.sender];
      balances[msg.sender] = 0;

     bool success =  erc20transfer(msg.sender, amount_to_transfer);
     require(success, "withdraw_amount error!");
}

//Modified verify_address_amount
function verify_address_amount(address mdd_address) public{
        uint256 re = erc20allowance(msg.sender,address(this));
        uint256 zt = find_address(mdd_address);
        if(zt != 999999999)
        {
            require(
            re >= address_info[zt].price,
            "Not enough amount"
            );

            bool success = erc20transferFrom(msg.sender,address(this),re);
            (success, "transfer error!");
            uint256 fee = re / 100 * 20;
            re = re - fee;

            balances[mdd_address] += re;
            balances[rev] += fee;

            address add = msg.sender;
            verify_info memory vvi;
            vvi.gm_address = add;
            vvi.dt = uint256(block.timestamp);
            vvi.price = re;

            address_info[zt].vi.push(vvi);
            address_info[zt].sl += 1;
        }
        else{
            require(
            re >= baseprice,
"Not enough amount"
            );

            bool success = erc20transferFrom(msg.sender,address(this),re);
            (success, "transfer error!");
            uint256 fee = re / 100 * 20;
            re = re - fee;

            balances[mdd_address] += re;
            balances[rev] += fee;

            uint256 asl = address_sz.length;
            address_sz.push(mdd_address);
            address add = msg.sender;
            address_info[asl].add = mdd_address;
            address_info[asl].price = baseprice;
            verify_info memory vvi;
            vvi.gm_address = add;
            vvi.dt = uint256(block.timestamp);
            vvi.price = re;

            address_info[asl].vi.push(vvi);
            address_info[asl].sl += 1;
        }
    }
```

Client Response

Contracts do not require this feature

GML-16:Precision issue in `GameLand_verify::verify_address_amount`

Category	Severity	Code Reference	Status	Contributor
Logical	Low	• code/payment_contract.sol#L120 • code/payment_contract.sol#L144	Fixed	w2ning, 0xzoobi

Code

```
120:            uint256 fee = re / 100 * 20;

144:            uint256 fee = re / 100 * 20;
```

Description

w2ning : Performing division before multiplication can lead to precision loss.

```
// Divide before multiply
uint256 fee = re / 100 * 20;
```

0xzoobi : Solidity's integer division can be truncated. As a result, precision loss can be precented by multiplying before dividing.
The current issue is on the fee calculation step wherein division is performed first and then multiplication. The impact does not seem to a severe one since `re` stores the allowances of a token. and it may only impact when `re < 100`. but it is a good practice to make sure you multiply before divide.

Recommendation

w2ning : Performing multiplication before division can sometimes avoid loss of precision.
Consider below fix in the `GameLand_verify.verify_address_amount()` function

```
// fix: Performing multiplication before division can sometimes avoid loss of precision
uint256 fee = re * 20 / 100;
```

0xzoobi : Consider multiplication before division to ensure precision in results.

Sample Fix:

```
uint256 fee = re * 20 / 100;
```

Reference - https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply

Client Response

Complete the modification

GML-17:Redundant use of receive and fallback in `GameLand_verify`

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/payment_contract.sol#L7-L9	Declined	0xzoobi

Code

```
7:    fallback() external payable {}
8:
9:    receive() external payable {}
```

Description

0xzoobi : The `receive` is used when a contract wants to receive ether and `fallback` is used for the same purpose but it also accepts calldata.

Using both of them may be required for a condition shown below but in the current scenario using `fallback` is sufficient.

```
fallback() external payable {
    result = doTask1(msg.data);
}

receive() external payable {
    doTask2();
}
```

Recommendation

0xzoobi : Remove the `receive()` function.

Client Response

The contract needs to be this way

GML-18:Reentrancy risk in `GameLand_verify` contract `verify_address_amount` function

Category	Severity	Code Reference	Status	Contributor
Reentrancy	Informational	• code/payment_contract.sol#L108-L166	Fixed	SAir

Code

```solidity
    function verify_address_amount(address mdd_address) public{
        uint256 re = erc20allowance(msg.sender,address(this));
        uint256 zt = find_address(mdd_address);
        if(zt != 999999999)
        {
            require(
            re >= address_info[zt].price,
            "Not enough amount"
            );

            bool success = erc20transferFrom(msg.sender,address(this),re);
            (success, "transfer error!");
            uint256 fee = re / 100 * 20;
            re = re - fee;
            bool success2 =  erc20transfer(mdd_address, re);
            require(success2, "transfer2 error!");
            bool success3 =  erc20transfer(rev, fee);
            require(success3, "transfer3 error!");

            address add = msg.sender;
            verify_info memory vvi;
            vvi.gm_address = add;
            vvi.dt = uint256(block.timestamp);
            vvi.price = re;

            address_info[zt].vi.push(vvi);
            address_info[zt].sl += 1;
        }
        else{
            require(
            re >= baseprice,
            "Not enough amount"
            );

            bool success = erc20transferFrom(msg.sender,address(this),re);
            (success, "transfer error!");
            uint256 fee = re / 100 * 20;
            re = re - fee;
            bool success2 =  erc20transfer(mdd_address, re);
            require(success2, "transfer2 error!");
            bool success3 =  erc20transfer(rev, fee);
            require(success3, "transfer3 error!");

            uint256 asl = address_sz.length;
            address_sz.push(mdd_address);
            address add = msg.sender;
            address_info[asl].add = mdd_address;
            address_info[asl].price = baseprice;
            verify_info memory vvi;
            vvi.gm_address = add;
            vvi.dt = uint256(block.timestamp);
            vvi.price = re;

            address_info[asl].vi.push(vvi);
            address_info[asl].sl += 1;
        }


    }
```

Description

SAir : In the verify_address_amount function, there are multiple function calls, including transfer, adding verification information, etc. Since we do not know the content of the ERC20.sol contract, attackers may use these function calls to carry out reentrancy attacks.

Recommendation

SAir : You can add a locking mechanism at the beginning of the function to ensure that the function can only be executed once when it is called, for example:

```
bool locked = false;

modifier reentrancyGuard {
    require(!locked, "Reentrancy guard failed");
    locked = true;
    _;
    locked = false;
}

function verify_address_amount(address mdd_address) public reentrancyGuard {
    ...
}
```

Client Response

Complete the modification

GML-19:Unuse the latest solidity version

Category	Severity	Code Reference	Status	Contributor
Logical	Informational	code/payment_contract.sol#L2	Declined	0xzoobi

Code

```
2:pragma solidity ^0.8.0;
```

Description

0xzoobi : The project is using solidity version `0.8.0`

Recommendation

0xzoobi : Use one of the recent versions like `0.8.16` or later.

Client Response

Contracts do not require this feature

GML-20:Unused function: `GameLand_verify::erc20approve`

Category	Severity	Code Reference	Status	Contributor
Gas Optimization	Informational	• code/payment_contract.sol#L225	Declined	w2ning

Code

```
225:    function erc20approve(address to, uint256 value) internal returns (bool success)  {
```

Description

w2ning : Fucntion `erc20approve` never used in `GameLand_verify` contract.

Removing this function can save gas when deploying contracts

```
    function erc20approve(address to, uint256 value) internal returns (bool success)  {
        bytes memory callload;
        callload = abi.encodeWithSignature(
                "approve(address,uint256)",
                to,
                value
            );
        (success, ) = usdt.call(callload);
        return success;
    }
```

Recommendation

w2ning : Delete the `erc20approve` fucntion.

Client Response

The contract needs to be this way

GML-21:Using a state variable to track the balance instead of `address(this).balance`

Category	Severity	Code Reference	Status	Contributor
Logical	Low	• code/payment_contract.sol#L213	Declined	0xzoobi

Code

```
213:        return address(this).balance;
```

Description

0xzoobi : The contract uses `address(this).balance` to track the current ether balance in the contract. This is not a good practice. Ether which was accidently sent to the contract via `selfdestruct` will also be taken into account.

Recommendation

0xzoobi : Define a state variable to track the contract balances and update it whenever the contract receives new ether.

Client Response

The contract needs to be this way

GML-22:for loop unlimited number of iterations risk in `GameLand_verify` contract `find_address` and `batch_set_address_amount` function

Category	Severity	Code Reference	Status	Contributor
Dos	**Low**	• **code/payment_contract.sol#L66-L72** • **code/payment_contract.sol#L76-L100**	**Declined**	**SAir**

Code

```
66:        for(uint256 i=0;i
```

Description

SAir : If an array is very large and the number of loops is not limited in the for loop, it may cause gas exhaustion and the contract cannot be used normally.

Recommendation

SAir : When doing a for loop, set the upper limit of the number of loops. If the number exceeds the number, an error message or rollback will be returned.
Consider below fix in the `GameLand_verify.set_address_amount()` function

```
uint256 maxForNumber;
```

Client Response

Contracts do not require this feature

Disclaimer