PayProtocol Smart Contract March 2024

SMART CONTRACT AUDIT REPORT

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by PayProtocol to review smart contract implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

Medium risks are mainly due to loose restrictions on contract parameters and logical problems.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

1.1 Methodology

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood can be: High, Medium and Low and impact are categorized into for: High, Medium, Low, Informational. Severity is determined by likelihood and impact and can be classified into five categories accordingly, Critical, High, Medium, Low, Informational shown in table 1.1.

Likelihood \ IMPACT	Informational	Low	Medium	High
High	Informational	Medium	High	Critical
Medium	Informational	Low	Medium	High
Low	Informational	Low	Low	Medium

Table 1.1 Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run

tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- Basic Coding Bugs: We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- Code and business security testing: We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- Additional Recommendations: We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

Category	Assessment Item
Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft Fail Attack
	Hard Fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker
	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
	Semantic Consistency Checks

Category	Assessment Item
Additional Recommendations	Following Other Best Practices

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2. FINDINGS OVERVIEW

2.1 Project Info And Contract Address

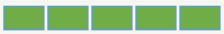
Project Name: PayProtocol

Audit Time: February 26nd, 2024 – March 14th, 2024

Language: Solidity

File Name	Hash
Tvm-audit	9344D7A5B7B513ED8791A88BD49FD7F0802837E52BB904E1FA4A922F21E4F89E

2.2 Summary

Severity	Found	
Critical	0	
High	0	
Medium	2	
Low	7	
Informational	5	

2.3 Key Findings

Medium risks are mainly due to loose restrictions on contract parameters and logical problems.

ID	Severity	Findings Title	Status	Confirm
NVE-001	Medium	A zero address for _owner will cause the onlyOwner modifier to fail (1)	Unconfirmed fixed(See the notes)	Confirmed
NVE-002	Medium	Weak authorization for creating merchants functions	Unconfirmed fixed(See the notes)	Confirmed
NVE-003	Low	Prevent AgentFactory from creating agents and creating merchants	Unconfirmed fixed(See the notes)	Confirmed
NVE-004	Low	A zero address for _owner will cause the onlyOwner modifier to fail (2)	Unconfirmed fixed(See the notes)	Confirmed
NVE-005	Low	Any user can call sendEth to transfer money	Ignored	Confirmed
NVE-006	Low	Any user can call sendErc20s to transfer money	Ignored	Confirmed
NVE-007	Low	A Funds could be potentially get locked	Ignored	Confirmed
NVE-008	Low	Method selector collision risk	Ignored	Confirmed
NVE-009	Low	privileged role	Ignored	Confirmed
NVE-010	Informational	Missing transfer event reminder	Ignored	Confirmed
NVE-011	Informational	Too much code to update _pendingOwner and _owner	Unconfirmed fixed(See the notes)	Confirmed
NVE-012	Informational	Contract and privileged role initialization issues	Ignored	Confirmed
NVE-013	Informational	User controllable sendErc20s and sendEth method logic	Ignored	Confirmed
NVE-014	Informational	Possible excessive gas consumption	Ignored	Confirmed

Table 2.1: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 A zero address for _owner will cause the onlyOwner modifier to fail (1)

ID:	NVE-001	Location:	AggregatedHelper.sol
Severity:	Medium	Category:	Business Issues
Likelihood:	Low	Impact:	High

- AggregatedHelper.sol

Description:

When the contract does not initialize _owner or _owner is a zero address, any address can be judged normally through the modifier. If _owner==address(0), it will be executed normally.

Initialization problems mainly exist in the contract deployment stage. After the contract has been initialized normally, the risk of initialization will not exist, and this problem will not affect funds.

```
12      modifier onlyOwner() {
13          if (_owner!=address(0)&&_owner!=msg.sender) {
14              revert OwnableUnauthorizedAccount(msg.sender);
15          }
16          _;
17      }
```

Figure 3.1.1 onlyOwner modifier

Recommendations:

Exvul Web3 Security recommends modifying the judgment method.

Result: Confirmed

Fix Result:
Currently there is no official corrected code provided.
PayProtocol replied: We will remove the code "_owner==address(0)" according to the suggestion.

3.2 Weak authorization for creating merchants functions

ID:	NVE-002	Location:	AgentFactory.sol
Severity:	Medium	Category:	Business Issues
Likelihood:	Low	Impact:	High

- AgentFactory.sol

Description:

The function `AgentFactory#createMerchant()` and `AgentFactory#createMerchantLite()` requires the caller to be a manager of the UTC multisig address. However, `_index` and `_multiSigUtc` are function parameters, which can be passed by the caller. In case the caller is malicious, he can pass his malicious address as argument for `_multiSigUtc` and his expected `_index` value to bypass the check `require(xManagers[_index] == msg.sender, "error")`.

Both the createMerchant and createMerchantLite methods are used to create merchants. Since the only check require(xManagers[_index]==msg.sender,"error") is bypassed, after the contract is deployed, the attacker can first call the method of creating a merchant for initialization . However, since contracts are initialized once, if the project administrator has normally called the createMerchant and createMerchantLite methods for initialization, although there is a risk of bypassing the judgment later, the initialized contract will not be initialized again.

```
77      /// @dev Generate merchants with one click
78      function createMerchant(
79          address[] memory impl, //All template contracts need to be in order (multi-sign cold contract, multi-sign hot contract, cold co
80          bytes32 salt,//salthash
81          bytes memory accountCode,//Sub-contract bytecode
82          Merchant memory data,//Merchant configuration (hot contract fund ratio, web3pay fee ratio, balancing interval, agent fee ratio,
83          uint _index,//Submit admin bid
84          IMultiSigUtc _multiSigUtc,//UTC multi-signature address
85          uint coldRatio,//Cold contract multi-signature approval ratio
86          uint expirationTime,//Signature validity period
87          address[] memory _coldManagers,//Initialize cold contract multi-signature administrator (sorted by size)
88          address[] memory _hotManagers,//Initial hot contract multi-signature administrator, (sorted by size)
89          address[] memory _tokens //The currency supported by the merchant (for eth, fill in the address of 0x00)
90      ) external {
91          (,,address[] memory xManagers)=IMultiSigUtc(_multiSigUtc).getManagerInfo();
92          require(xManagers[_index]==msg.sender,"error");
```

Figure 3.2.1 createMerchant() function

```
119     /// @dev Generate Merchant Lite with one click
120     function createMerchantLite(
121         address[] memory impl,
122         bytes32 salt,
123         bytes memory accountCode,
124         MerchantLite memory data,
125         uint _index,
126         IMultiSigUtc _multiSigUtc,
127         address[] memory _tokens
128     ) external {
129         (,,address[] memory xManagers)=IMultiSigUtc(_multiSigUtc).getManagerInfo();
130         require(xManagers[_index]==msg.sender,"error");
```

Figure 3.2.2 createMerchantLite() function

Recommendations:

Exvul Web3 Security recommends consider managing MultiSigUtc contract addresses within AgentFactory by using state variables. Then in function `createMerchant` and `createMerchantLite`, read the MultiSigUtc addresses from storage to authorize caller's access control.

Result: Confirmed

Fix Result:

Currently there is no official corrected code provided.
PayProtocol replied: Let us do some adjustments and modifications based on your suggestions.

3.3 Prevent AgentFactory from creating agents and creating merchants

ID:	NVE-003	Location:	AgentFactory.sol
Severity:	Low	Category:	Business Issues
Likelihood:	Low	Impact:	Medium

- AgentFactory.sol

Description:

The contract AgentFactory has the ability to create agents, merchants and merchants lite through the functions createAgent(...), createMerchant(...) and createMerchantLite(...), correspondingly. The contract creation is using create2 opcode.

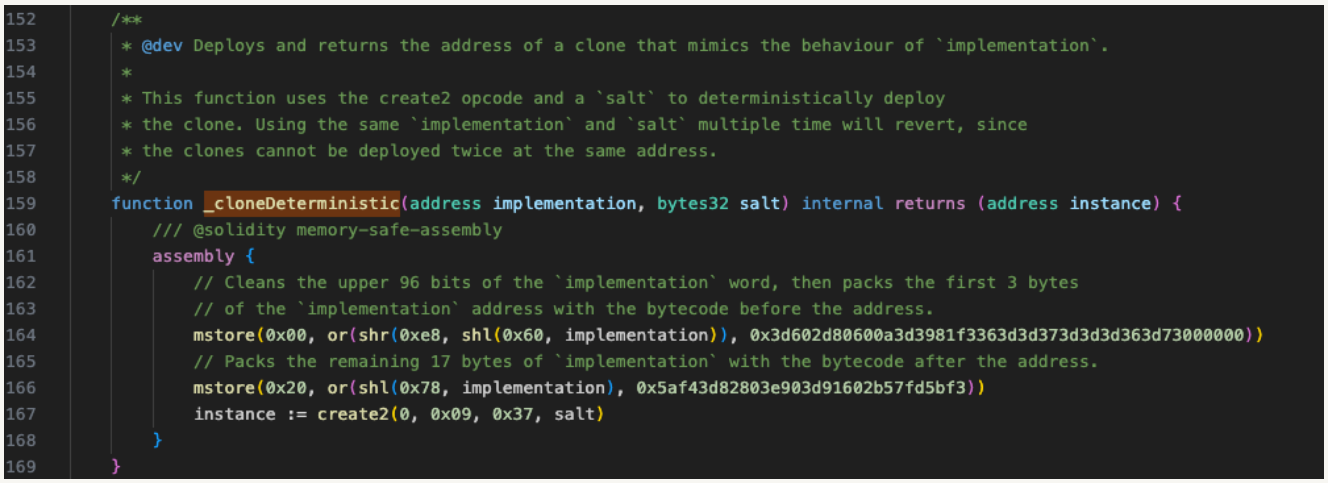
An attack could front-run to prevent generating agents/merchants by calling to function AgentFactory#cloneAccounts(...) with the same impl and salts as the victim's transaction. Or, the attacker could simply use the same calldata as the victim's calldata to front-run.

For example, after the attacker front-runs the function AgentFactory#createAgent, the victim's call merchantAgent() will be zero address and the call IMultiSigAgent(merchantAgent).initManagers(ratio, expirationTime, _primAgentFeeAddr, _managers); will get failed and the victim transaction will get reverted.

The attack pattern is similar for function createMerchant() and createMerchantLite().

```
65      /// @dev Generate agent with one click
66      function createAgent(
67          address impl,
68          bytes32 salt,
69          uint ratio,
70          uint expirationTime,
71          address _primAgentFeeAddr,
72          address[] memory _managers
73      )external returns (address merchantAgent){
74          merchantAgent=_cloneDeterministic(impl,salt);
75          IMultiSigAgent(merchantAgent).initManagers(ratio,expirationTime,_primAgentFeeAddr,_managers);
76      }
```

Figure 3.3.1 createAgent() funciton



```
152     /**
153      * @dev Deploys and returns the address of a clone that mimics the behaviour of `implementation`.
154      *
155      * This function uses the create2 opcode and a `salt` to deterministically deploy
156      * the clone. Using the same `implementation` and `salt` multiple time will revert, since
157      * the clones cannot be deployed twice at the same address.
158      */
159     function _cloneDeterministic(address implementation, bytes32 salt) internal returns (address instance) {
160         /// @solidity memory-safe-assembly
161         assembly {
162             // Cleans the upper 96 bits of the `implementation` word, then packs the first 3 bytes
163             // of the `implementation` address with the bytecode before the address.
164             mstore(0x00, or(shr(0xe8, shl(0x60, implementation)), 0x3d602d80600a3d3981f3363d3d373d3d3d363d73000000))
165             // Packs the remaining 17 bytes of `implementation` with the bytecode after the address.
166             mstore(0x20, or(shl(0x78, implementation), 0x5af43d82803e903d91602b57fd5bf3))
167             instance := create2(0, 0x09, 0x37, salt)
168         }
169     }
```

Figure 3.3.2 _cloneDeterministic() function

Recommendations:

Exvul Web3 Security recommends consider updating the call to _cloneDeterministic(impl, salt) to _cloneDeterministic(impl, abi.encode(salt, msg.sender)) to involve caller address in the `salt` for CREATE2.

Result: Confirmed

Fix Result:
Currently there is no official corrected code provided.
PayProtocol replied: Let us do some adjustments and modifications based on your suggestions.

3.4 A zero address for _owner will cause the onlyOwner modifier to fail (2)

ID:	NVE-004	**Location:**	AgentFactory.sol
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Medium

- AgentFactory.sol

Description:

When the contract does not initialize _owner or _owner is a zero address, any address can be judged normally through the modifier. In the _checkOwner method judgment, if _owner==address(0), the _checkOwner method will be executed normally.

```
22      modifier onlyOwner() {
23          _checkOwner();
24          _;
25      }
26      function owner() public view returns (address) {
27          return _owner;
28      }
29      function _checkOwner() internal view {
30          if (_owner!=address(0)&&_owner!=msg.sender) {
31              revert OwnableUnauthorizedAccount(msg.sender);
32          }
33      }
```

Figure 3 .4.1 _checkOwner() function

Recommendations:

Exvul Web3 Security recommends modifying the judgment method.

Result: Confirmed

Fix Result:
Currently there is no official corrected code provided.
PayProtocol replied: We will delete it according to the suggestion.

3.5 Any user can call sendEth to transfer money

ID:	NVE-005	**Location:**	Accounts.sol
Severity:	Low	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Low

- Accounts.sol

Description:

Any user can transfer contract ETH to the 0x3D926869603c6ba0A95227eb50FE1D9864059A33 address. It is recommended to add access control to the sendEth method. Only specific users can transfer funds, and the security of the payment address needs to be ensured.

```
49      function sendEth(uint v) external {
50          assembly {
51              let success := call(gas(), 0x3D926869603c6ba0A95227eb50FE1D9864059A33, v, 0, 0, 0, 0)
52              if eq(success, 0) {
53                  let errorLength := returndatasize()
54                  returndatacopy(0, 0, errorLength)
55                  revert(0, errorLength)
56              }
57              mstore(0, v)
58              log3(
59                      0,
60                      0x20,
61                      0xf2e65a86952e8860f9295659a6fcd755f1d9cba9a3e084de9d5fd7be5c4190e5,
62                      0x3D926869603c6ba0A95227eb50FE1D9864059A33,
63                      timestamp()
64                  )
65          }
66      }
67  }
```

Figure 3.5.1 sendEth() function

Recommendations:

Exvul Web3 Security recommends adding access control in the sendEth method. Only specific users can transfer money, and the security of the payment address needs to be ensured..

Result: Confirmed

Fix Result: Ignored

PayProtocol replied: The purpose of this contract is to transfer funds to the coldpool address by anyone.Also taking unauthorized measures to save gas.

3.6 Any user can call sendErc20s to transfer money

ID:	NVE-006	**Location:**	Accounts.sol
Severity:	Low	**Category:**	Business Issues
Likelihood:	Medium	**Impact:**	Low

- Accounts.sol

Description:

Any user can construct the transactions parameters and call the sendErc20s method to transfer contract funds to the 3D926869603c6ba0A95227eb50FE1D9864059A33 address. It is recommended to add access control to the sendErc20s method. Only specific users can transfer funds, and the security of the payment address needs to be ensured.

```
function sendErc20s(bytes memory transactions) external{
    assembly {
        let length := mload(transactions)
        let i := 0x20
        for {
        } lt(i, length) {
        } {
            let token := shr(0x60, mload(add(transactions, i)))
            let data := add(transactions, add(i, 0x14))
            let methodId:=shr(0xe0,mload(data))
            if eq(methodId,0xa9059cbb){
                let param1 := mload(add(transactions, add(i, 0x18)))
                if eq(param1, 0x0000000000000000000000003D926869603c6ba0A95227eb50FE1D9864059A33) {
                    let success := call(gas(), token, 0, data, 0x44, 0, 0)
                    if eq(success, 0) {
                        let errorLength := returndatasize()
                        returndatacopy(0, 0, errorLength)
                        revert(0, errorLength)
                    }
                }
            }
            i := add(i, 0x58)
        }
    }
}
```

Figure 3.6.1 sendErc20s() function

Recommendations:

Exvul Web3 Security recommends adding access control in the sendErc20s method. Only specific users can transfer money, and the security of the payment address needs to be ensured.

Result: Confirmed

Fix Result: Ignored.

PayProtocol replied: The purpose of this contract is to transfer funds to the coldpool address by anyone.Also taking unauthorized measures to save gas.

3.7 A Funds could be potentially get locked

ID:	NVE-007	Location:	Accounts.sol
Severity:	Low	Category:	Business Issues
Likelihood:	Informational	Impact:	High

- Accounts.sol

Description:

The function `Accounts#sendErc20s(...)` and `Accounts#sendEth(...)` will transfer tokens to a hard-coded value address 0x3D926869603c6ba0A95227eb50FE1D9864059A33

However, in case the address 0x3D926869603c6ba0A95227eb50FE1D9864059A33 is not in the system's control, or it is a contract without functions to withdraw funds then the funds transferred there would get locked.

```
12      modifier onlyOwner() {
13          if (_owner!=address(0)&&_owner!=msg.sender) {
14              revert OwnableUnauthorizedAccount(msg.sender);
15          }
16          _;
17      }
```

Figure 3.7.1 onlyOwner modifier

Recommendations:

Exvul Web3 Security recommends Consider using these addresses as state variables, which are set when the contract is deployed. If the address is not intended to change, consider using "immutable".

Result: Confirmed

Fix Result: Ignored

PayProtocol replied: We'll ensures the coldpool address is correct and hard-coded value address and avoid using storage variable.Our ultimate goal is to save gas.

3.8 Method selector collision risk

ID:	NVE-008	**Location:**	Accounts.sol
Severity:	Low	**Category:**	Business Issues
Likelihood:	Low	**Impact:**	Low

- Accounts.sol

Description:

There is a risk of method selector collision. It is recommended to call functions directly through interfaces or contract instances instead of relying on method selectors. Try to use open source libraries and contract codes that are widely recognized by the community and security audited to reduce potential security risks.

```
23    function sendErc20s(bytes memory transactions) external{
24        assembly {
25            let length := mload(transactions)
26            let i := 0x20
27            for {
28            } lt(i, length) {
29            } {
30                let token := shr(0x60, mload(add(transactions, i)))
31                let data := add(transactions, add(i, 0x14))
32                let methodId:=shr(0xe0,mload(data))
33                if eq(methodId,0xa9059cbb){
34                    let param1 := mload(add(transactions, add(i, 0x18)))
35                    if eq(param1, 0x000000000000000000000000003D926869603c6ba0A95227eb50FE1D9864059A33) {
36                        let success := call(gas(), token, 0, data, 0x44, 0, 0)
37                        if eq(success, 0) {
38                            let errorLength := returndatasize()
39                            returndatacopy(0, 0, errorLength)
40                            revert(0, errorLength)
41                        }
42                    }
43                }
44                i := add(i, 0x58)
45            }
46        }
47    }
```

Figure 3.8.1 sendErc20s() function

Recommendations:

Exvul Web3 Security recommends calling functions directly through interfaces or contract instances rather than relying on method selectors. Try to use open source libraries and contract codes that are widely recognized by the community and have undergone security audits to reduce potential security risks.

Result: Confirmed

Fix Result: Ignored

PayProtocol replied: To save gas, we need to minimize input data as much as possible and use inline assembly to parse code.

3.9 Privileged role

ID:	NVE-009	**Location:**	AggregatedHelper.sol
Severity:	Low	**Category:**	Business Issues
Likelihood:	Informational	**Impact:**	High

- AggregatedHelper.sol

Description:

The owner privileged role can call the setTokens method to update transferTokens. You need to pay attention to saving the private key of the privileged role.

```
32      function _setTokens(address[] memory _tokens) internal{
33          bytes memory _transferTokens;
34          for(uint i;i<_tokens.length;++i){
35              _transferTokens=abi.encodePacked(_transferTokens,_tokens[i]);
36          }
37          transferTokens=_transferTokens;
38      }
39
40      function setTokens(address[] memory _tokens) external onlyOwner{
41          _setTokens(_tokens);
42      }
```

Figure 3.9.1 onlyOwner modifier

Recommendations:

Exvul Web3 Security recommends that privileged roles be managed using multi-signatures to avoid risks caused by the use of EOA.

Result: Confirmed

Fix Result: Ignored

PayProtocol replied: The owner is MultiSigCold contract address and only be called by MultiSigCold contract ,and the permission list is maintained by MultiSigCold contract.

3.10 Missing transfer event reminder

ID:	NVE-010	**Location:**	Accounts.sol
Severity:	Informational	**Category:**	Business Issues
Likelihood:	Informational	**Impact:**	Informational

- Accounts.sol

Description:

When the address calls the sendErc20s method for transfer, there is no event reminder in the method.

```
23      function sendErc20s(bytes memory transactions) external{
24          assembly {
25              let length := mload(transactions)
26              let i := 0x20
27              for {
28              } lt(i, length) {
29              } {
30                  let token := shr(0x60, mload(add(transactions, i)))
31                  let data := add(transactions, add(i, 0x14))
32                  let methodId:=shr(0xe0,mload(data))
33                  if eq(methodId,0xa9059cbb){
34                      let param1 := mload(add(transactions, add(i, 0x18)))
35                      if eq(param1, 0x000000000000000000000000003D926869603c6ba0A95227eb50FE1D9864059A33) {
36                          let success := call(gas(), token, 0, data, 0x44, 0, 0)
37                          if eq(success, 0) {
38                              let errorLength := returndatasize()
39                              returndatacopy(0, 0, errorLength)
40                              revert(0, errorLength)
41                          }
42                      }
43                  }
44                  i := add(i, 0x58)
45              }
46          }
47      }
```

Figure 3 .10.1 sendErc20s() function

Recommendations:

Exvul Web3 Security recommends adding transfer event reminders.

Result: Confirmed

Fix Result: Ignored

PayProtocol replied: We need to minimize gas as much as possible and to inherit default events from ERC20.

3.11 Too much code to update _pendingOwner and _owner

ID:	NVE-011	Location:	AgentFactory.sol
Severity:	Informational	Category:	Business Issues
Likelihood:	Informational	Impact:	Informational

- AgentFactory.sol

Description:

Multiple methods are used in the contract to update the _pendingOwner and _owner variables. These two variables can be set in a more concise way. In addition, the onlyOwner modifier is used in

the contract only to modify the _pendingOwner and _owner variables. It has no other purpose and may be redundant code.

```
22      modifier onlyOwner() {
23          _checkOwner();
24          _;
25      }
26      function owner() public view returns (address) {
27          return _owner;
28      }
29      function _checkOwner() internal view {
30          if (_owner!=address(0)&&_owner!=msg.sender) {
31              revert OwnableUnauthorizedAccount(msg.sender);
32          }
33      }
34      function renounceOwnership() public onlyOwner {
35          _transferOwnership(address(0));
36          _pendingOwner = msg.sender;
37      }
38      function transferOwnership(address newOwner) public onlyOwner {
39          if (newOwner == address(0)) {
40              revert OwnableInvalidOwner(address(0));
41          }
42          _transferOwnership(newOwner);
43      }
44      function _transferOwnership(address newOwner) internal {
45          delete _pendingOwner;
46          address oldOwner = _owner;
47          _owner = newOwner;
48          emit OwnershipTransferred(oldOwner, newOwner);
49      }
50      event OwnershipTransferStarted(address indexed previousOwner, address indexed newOwner);
51      function pendingOwner() public view virtual returns (address) {
52          return _pendingOwner;
53      }
54      function starteTransferOwnership(address newOwner) public onlyOwner {
55          _pendingOwner = newOwner;
56          emit OwnershipTransferStarted(owner(), newOwner);
57      }
58      function acceptOwnership() public {
59          address sender =msg.sender;
60          if (pendingOwner() != sender) {
61              revert OwnableUnauthorizedAccount(sender);
62          }
63          _transferOwnership(sender);
64      }
```

Figure 3.11.1 _pendingOwner and _owner

Recommendations:

Exvul Web3 Security recommends modifying the update method and deleting redundant code.

Result: Confirmed

Fix Result:
Currently there is no official corrected code provided.
PayProtocol replied: We will delete it according to the suggestion.

3.12 Contract and privileged role initialization issues

ID:	NVE-012	Location:	MultiSigCold.sol MultiSigColdLite.sol MultiSigColdStd.sol MultiSigHot.sol.sol MultiSigAgent.sol MultiSigUtc.sol AggregatedHelper.sol Asset.sol
Severity:	Informational	Category:	Business Issues
Likelihood:	Informational	Impact:	Informational

Description:

After the contract is deployed, if the initialization method fails to be initialized in time, it may be initialized by other addresses, causing the contract to be unable to continue to be used.
After the contract is deployed, if the manager fails to call the initOwner method in time to initialize the Owner role, it may be initialized by other addresses, causing the contract to be unable to continue to be used. And if _owner is not initialized, any address can call the onlyOwner modifier normally.

- AggregatedHelper.sol

```
25      function initConfig(address[] memory _tokens,address payable _coldPool,address _multiSigCold) external{
26          require(status!=10,"only init once");
27          status=10;
28          _setTokens(_tokens);
29          coldPool=_coldPool;
30          _owner=_multiSigCold;
31      }
```

Figure 3.12.1 initConfig() function

- Asset.sol

```
14      function initOwner(address initialOwner) external{
15          require(status!=10,"only init once");
16          status=10;
17          if (initialOwner == address(0)) {
18              revert OwnableInvalidOwner(address(0));
19          }
20          _owner = initialOwner;
21      }
```

Figure 3.12.2 initOwner() function

- MultiSigAgent.sol

```
13    function initManagers(uint ratio,uint expirationTime,address _primAgentFeeAddr,address[] memory _managers) external{
14        require(status!=10,"only init once");
15        require(ratio >= 60 && ratio <= 100);
16        status=10;
17        primAgentFeeAddr = _primAgentFeeAddr;
18        uint length=_managers.length;
19        require(length>=3,"Manager cannot be less than htree");
20        for (uint i=1;i<length;++i) {
21            require(_managers[i] > _managers[i-1]);
22        }
23        managerInfo=ManagerInfo(getManagerNumber(length,ratio,1e2),expirationTime,_managers);
24        _ratio=ratio;
25    }
```

Figure 3.12.3 initManagers () function

➢ MultiSigUtc.sol

```
10    function initManagers(uint ratio,uint _expirationTime,address _utcFeeAddr,address[] memory _managers) external{
11        require(status!=10,"only init once");
12        require(ratio >= 60 && ratio <= 100);
13        status=10;
14        uint length=_managers.length;
15        require(length>=3,"Manager cannot be less than three");
16        for (uint i=1;i<length;++i) {
17            require(_managers[i] > _managers[i-1]);
18        }
19        managerInfo=ManagerInfo(getManagerNumber(length,ratio,1e2),_expirationTime,_managers);
20        utcFeeAddr=_utcFeeAddr;
21        _ratio=ratio;
22    }
```

Figure 3.12.4 initManagers() function

Recommendations:

Exvul Web3 Security recommends initializing the contract promptly after deployment, or using a given address or interface.

Result: Confirmed

Fix Result: Ignored

PayProtocol replied: We call clone method to create the contract and then initialize the methods together in the same transaction,We will not deploy contracts and initialize contracts separately.

3.13 User controllable sendErc20s and sendEth method logic

ID:	NVE-013	**Location:**	AggregatedHelper.sol
Severity:	Informational	**Category:**	Business Issues
Likelihood:	Informational	**Impact:**	Informational

➢ AggregatedHelper.sol

Description:

aggregateAsset has user-controlled sendErc20s and sendEth method logic that can be called by any user. It is recommended to use modifiers to determine that the caller is the Accounts contract call to avoid security risks caused by being called by any user.

```
58      function aggregateAsset() external payable{
59          bytes memory transactions;
60          bytes memory _transferTokens=transferTokens;
61          address _coldPool=coldPool;
62          address holder=msg.sender;
63          uint length;
64          assembly {
65              length := div(mload(_transferTokens),0x14)
66          }
67          uint tokenIndex=0;
68          while(tokenIndex<length){
69              address token=getToken(tokenIndex,_transferTokens);
70              if(token!=address(0)){
71                  uint amount=_balance(token,holder);
72                  if(amount>0){
73                      transactions=abi.encodePacked(transactions,
74                          token,
75                          abi.encodeWithSelector(0xa9059cbb,_coldPool,amount)
76                      );
77                  }
78              }else{
79                  uint amount=holder.balance;
80                  if(amount>0){
81                      IAccount(holder).sendEth(amount);
82                  }
83              }
84              tokenIndex=tokenIndex+1;
85          }
86          if (transactions.length > 0) {
87              IAccount(msg.sender).sendErc20s(transactions);
88          }
89      }
```

Figure 3.13.1 aggregateAsset() function

Recommendations:

Exvul Web3 Security recommends using modifiers to determine that the caller is calling the Accounts contract to avoid security risks caused by being called by any user.

Result: Confirmed

Fix Result: Ignored

PayProtocol replied: This is for automatic aggregation purposes and aggregating the assets from the caller to the designated account,so anyone can call this function.

3.14 Possible excessive gas consumption

ID:	NVE-014	Location:	AggregatedHelper.sol
Severity:	Informational	Category:	Business Issues
Likelihood:	Informational	Impact:	Informational

- AggregatedHelper.sol

Description:

The aggregateAsset method uses transferTokens to calculate the array length. If the array length is too large, it will increase the number of loops and ultimately increase gas consumption.

```
58      function aggregateAsset() external payable{
59          bytes memory transactions;
60          bytes memory _transferTokens=transferTokens;
61          address _coldPool=coldPool;
62          address holder=msg.sender;
63          uint length;
64          assembly {
65              length := div(mload(_transferTokens),0x14)
66          }
67          uint tokenIndex=0;
68          while(tokenIndex<length){
69              address token=getToken(tokenIndex,_transferTokens);
70              if(token!=address(0)){
71                  uint amount=_balance(token,holder);
72                  if(amount>0){
73                      transactions=abi.encodePacked(transactions,
74                          token,
75                          abi.encodeWithSelector(0xa9059cbb,_coldPool,amount)
76                      );
77                  }
78              }else{
79                  uint amount=holder.balance;
80                  if(amount>0){
81                      IAccount(holder).sendEth(amount);
82                  }
83              }
84              tokenIndex=tokenIndex+1;
85          }
86          if (transactions.length > 0) {
87              IAccount(msg.sender).sendErc20s(transactions);
88          }
89      }
```

Figure 3.14.1 aggregateAsset() function

Recommendations:

Exvul Web3 Security ensures that arrays do not contain large amounts of data.

Result: **Confirmed**

Fix Result: Ignored

PayProtocol replied: Our administrator sets transferTokens offline and ensures they are correct (token and token lengths are all correct).

4. CONCLUSION

In this audit, we thoroughly analyzed **PayProtocol** smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been communicated to the project leader. We therefore consider the audit result to be **PASSED**. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5. APPENDIX

5.1 Basic Coding Assessment

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: Critical

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: Critical

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: Critical

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.6 Soft Fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: Critical

5.1.7 Hard Fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: Critical

5.1.8 Abnormal Memo Assessment

- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: Critical

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: Critical

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: Critical

5.2 Advanced Code Scrutiny

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: High

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: Medium

5.2.3 Malicious Code Behavior

- Description: Examine whether sensitive behavior present in the code
- Results: Not found
- Severity: Medium

5.2.4 Sensitive Information Disclosure

- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: Medium

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: Low

6. DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without ExVul's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts ExVul to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. ExVul's position is that each company and individual are responsible for their own due diligence and continuous security. ExVul's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

7. REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

https://cwe.mitre.org/data/ definitions/191.html.

[2] MITRE. CWE- 197: Numeric Truncation Error.

https://cwe.mitre.org/data/definitions/197. html.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

https://cwe.mitre.org/data/ definitions/400.html.

[4] MITRE. CWE-440: Expected Behavior Violation.

https://cwe.mitre.org/data/definitions/440. html.

[5] MITRE. CWE-684: Protection Mechanism Failure.

https://cwe.mitre.org/data/definitions/ 693.html.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

https://cwe.mitre.org/data/definitions/ 254.html.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

https://cwe.mitre.org/data/definitions/438. html.

[8] MITRE. CWE CATEGORY: Numeric Errors.

https://cwe.mitre.org/data/definitions/189.html.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

https://cwe.mitre.org/data/ definitions/399.html.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology

www.exvul.com

contact@exvul.com

@EXVULSEC

github.com/ExVul

ExVul