OORT

BLOCKCHAIN AUDIT REPORT

September 2023

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by OORT to review Blockchain 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.

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.

	IMPACT			
Likelihood	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 Blockchain 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 Blockchains from the perspective of proven programming practices.

Category	Assessment Item
P2P Communication Security	Connection Number Occupation Audit
	Eclipse Attack
	Packet Size Limit
	Node Communication Protocol Security
RPC Interface Security	RPC Sensitive Interface Permissions
	Traditional Web Security
	RPC Interface Security
Consensus Mechanism Security	Design Of Consensus Mechanism
	Implementation Of Consensus Verification
	Incentive Mechanism Audit
Transaction processing Security	Transaction Signature Logic
	Transaction Verification Logic
	Transaction Processing Logic
	Transaction Fee Setting
	Transaction Replay
Cryptography Security	Random Number Range And Probability Distribution
	Cryptographic Algorithm Lmplementation/Use
Wallet Module & Account Security Audit	Private Key / Mnemonic Word Storage Security
	Private Key / Mnemonic Word Usage Security
	Private key/mnemonic generation algorithm
Others Security Audit	Database Security
	Thread Security
	File Permission Security

Category	Assessment Item
	Historical Vulnerability Security

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

Project Name: OORT

Audit Time: August 1, 2023 – September18, 2023

Language: C++

File Name	HASH
OORT	https://github.com/oort-tech/Olympus/commit/98b372d4f0878857849b21a161c5e2622544f65a

2.2 Summary

Severity	Found	
Critical	0	
High	2	🟥🟥
Medium	4	🟨🟨🟨🟨
Low	3	🟦🟦🟦
Informational	0	

2.3 Key Findings

ID	Severity	Findings Title	Status	Confirm
NVE-001	High	Integer Overflow in insert_column_families function	Fixed	Confirmed
NVE-002	Medium	Miss default case	Fixed	Confirmed
NVE-003	Medium	Integer Overflow at function rawV	Fixed	Confirmed
NVE-004	Medium	Node can be DoS if block does not exist.	Fixed	Confirmed
NVE-005	Medium	out-of-bounds access in GetStakingList	Fixed	Confirmed
NVE-006	Low	strcmp is not a safe string comparison function	Fixed	Confirmed
NVE-007	High	No lock when insert in the map object	Fixed	Confirmed
NVE-008	Low	Nullptr access may lead to Node DoS	Fixed	Confirmed
NVE-009	Low	count should not use sign type	Fixed	Confirmed

Table 2.3: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 Integer Overflow in insert_column_families function

ID:	NVE-001	**Location:**	mcp/db/column.cpp
Severity:	High	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	High

Description:

If push_back fails, the size of m_column_families will be 0, causing integer overflow.

```
int mcp::db::db_column::insert_column_families(std::string const& name,
std::shared_ptr<rocksdb::ColumnFamilyOptions> cfops)

{

    rocksdb::ColumnFamilyOptions faOption = *cfops;

    m_column_families.push_back(rocksdb::ColumnFamilyDescriptor(name, faOption));

    return m_column_families.size() - 1;

}
```

Result: Confirmed

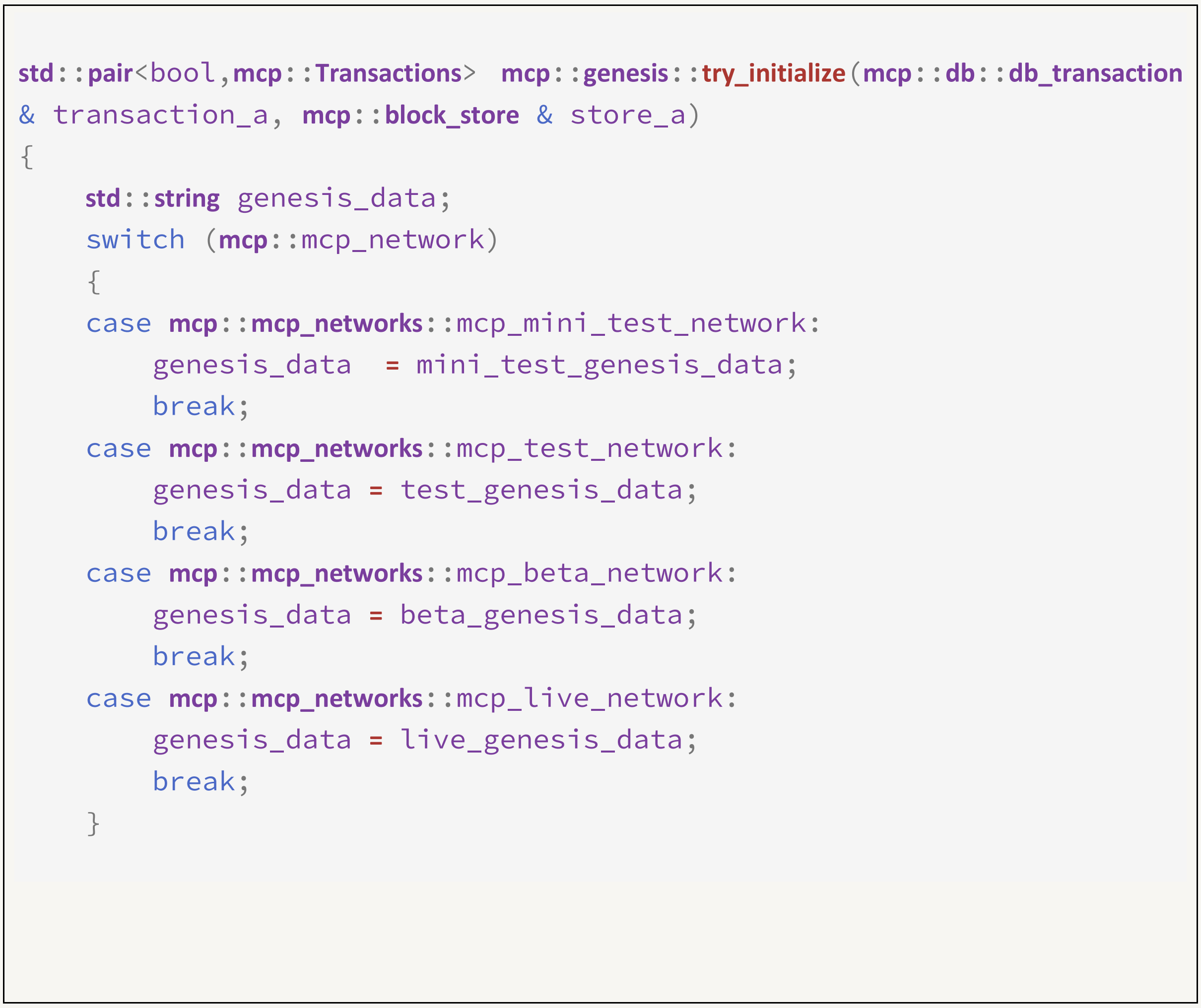
Fix Result: Fixed

3.2 Miss default case

ID:	NVE-002	**Location:**	mcp/core/genesis.cpp
Severity:	Medium	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	Medium

Description:

The function not concern the exception solution when network is no include in mcp_mini_test_network, mcp_test_network, mcp_beta_network, mcp_live_network,so the genesis_data will not initial, and then will lead to some serious problem.



```
std::pair<bool,mcp::Transactions>  mcp::genesis::try_initialize(mcp::db::db_transaction
& transaction_a, mcp::block_store & store_a)
{
    std::string genesis_data;
    switch (mcp::mcp_network)
    {
    case mcp::mcp_networks::mcp_mini_test_network:
        genesis_data  = mini_test_genesis_data;
        break;
    case mcp::mcp_networks::mcp_test_network:
        genesis_data = test_genesis_data;
        break;
    case mcp::mcp_networks::mcp_beta_network:
        genesis_data = beta_genesis_data;
        break;
    case mcp::mcp_networks::mcp_live_network:
        genesis_data = live_genesis_data;
        break;
    }
```

Result: Confirmed

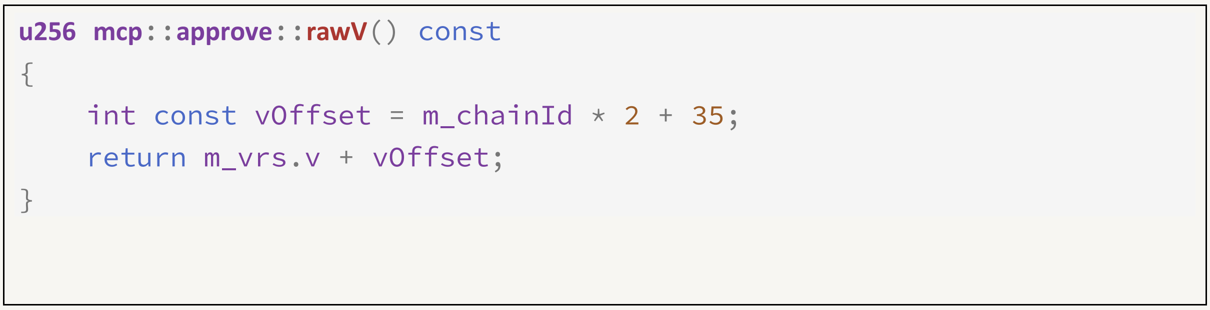
Fix Result: Fixed

3.3 Integer Overflow at function rawV

ID:	NVE-003	**Location:**	mcp/core/approve.cpp
Severity:	Medium	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	Medium

Description:

M_vrs.v is of byte type, which is uint8_t type. vOffset is of int type, which will cause integer overflow. The type of vOffset should be initialized to uint64_t.



```
u256 mcp::approve::rawV() const
{
    int const vOffset = m_chainId * 2 + 35;
    return m_vrs.v + vOffset;
}
```

Result: Confirmed

Fix Result: Fixed

3.4 Node can be DoS if block does not exist.

ID:	NVE-004	**Location:**	mcp/core/block_store.cpp
Severity:	Medium	**Category:**	P2P Communication Security
Likelihood:	Medium	**Impact:**	Medium

Description:

The block in line 9 may be empty. You need to check whether the block is empty. If it is empty, inserting blocks will cause more serious problems.

```
std::shared_ptr<mcp::block>       mcp::block_cache::block_get(mcp::db::db_transaction
&transaction_a, mcp::block_hash const &block_hash_a)
{
    std::shared_ptr<mcp::block> block;
    std::lock_guard<std::mutex> lock(m_block_mutex);
    if (!m_block_changings.count(block_hash_a))
    {
        bool exists = m_blocks.tryGet(block_hash_a, block);
        if (!exists)
            block = m_store.block_get(transaction_a, block_hash_a);

        m_blocks.insert(block_hash_a, block);
    }
    else
        block = m_store.block_get(transaction_a, block_hash_a);

    return block;
}
```

From block_get code we can see if the 'exists' is 0. Will lead to result equal to null.so from the above code, we can see we will insert a empty block if the block_get return nullptr.

```
std::shared_ptr<mcp::block>    mcp::block_store::block_get(mcp::db::db_transaction    &
transaction_a, mcp::block_hash const & hash_a)
{
    std::string value;
    bool      exists(transaction_a.get(blocks,      mcp::h256_to_slice(hash_a),
value));
```

```
    std::shared_ptr<mcp::block> result = nullptr;
    if (exists)
    {
        dev::RLP r(value);
       //auto mode = IncludeSignature::WithSignature;
       //if (hash_a == mcp::genesis::block_hash)
       //    mode = IncludeSignature::WithoutSignature;

        result = std::make_shared<mcp::block>(r);
        assert_x_msg(result != nullptr, "hash:" + hash_a.hex() + " ,data:"
+ value);
    }
    return result;
}
```

Result: Confirmed

Fix Result: Fixed

3.5 out-of-bounds access in GetStakingList

ID:	NVE-005	**Location:**	mcp/core/block_store.cpp
Severity:	Medium	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	Medium

Description:

The length of _r should be judged to avoid out-of-bounds access,this code access _r[1],if _r's length small than 2 will lead to out of bound access.

```
mcp::StakingList mcp::block_store::GetStakingList(mcp::db::db_transaction &
_transaction, Epoch const & _epoch)
{
    mcp::StakingList ret;
    std::string value;
    bool exists(_transaction.get(stakingList,
mcp::h64_to_slice(h64(_epoch)), value));
    if (exists)
    {
        dev::RLP r(value);
        assert_x(r.isList());
        for (dev::RLP _r : r)
        {
            auto _a = (dev::Address)_r[0];
            auto _b = _r[1].toInt<dev::u256>();
            ret[_a] = _b;
        }
    }
    return ret;
}
```

Result: Confirmed

Fix Result: Fixed

3.6 strcmp is not a safe string comparison function

ID:	NVE-006	**Location:**	mcp/rpc/rpc_ws.cpp
Severity:	Low	**Category:**	Transaction processing Security
Likelihood:	Low	**Impact:**	Low

Description:

If the message is a null pointer, it may cause node DOS. It is recommended to use the strcmp_s function.

```
int mcp::subscribe::get_index_by_message(std::string message)
{
    int ret = 0;
    rLock lock(mutex);

    std::map<int, std::string>::iterator itsub = subcribe_list.begin();
    for (; itsub != subcribe_list.end(); itsub++)
    {
        if (strcmp(itsub->second.c_str(), message.c_str()) == 0)
        {
            ret = itsub->first;
        }
    }
    return ret;
}
```

Result: Confirmed

Fix Result: Fixed

3.7 No lock when insert in the map object

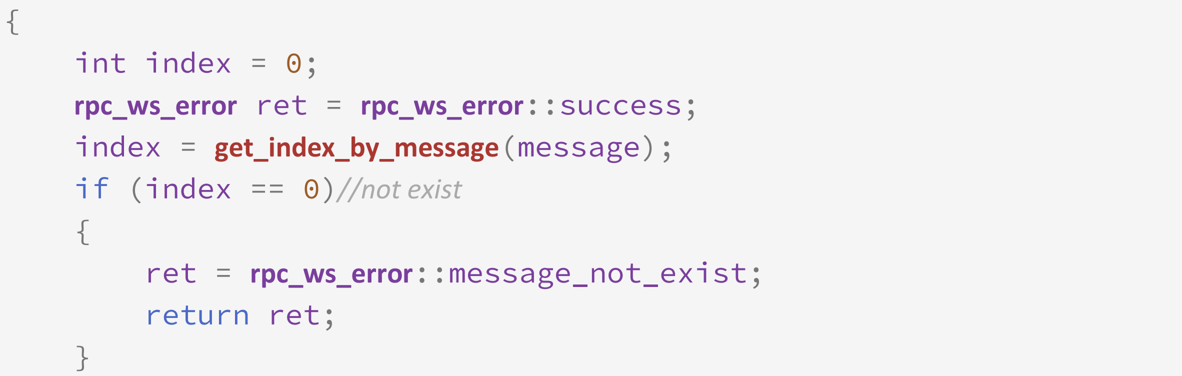
ID:	NVE-007	**Location:**	mcp/rpc/rpc_ws.cpp
Severity:	High	**Category:**	Transaction processing Security
Likelihood:	Medium	**Impact:**	High

Description:

This function is not use locked. Subcribe_list should be locked when inserted to prevent conditional competition vulnerabilities from occurring and causing the possibility of code execution.Colleagues, get_max_index() should be restricted to be less than 0xffffffff. If get_max_index() is equal to 0xffffffff, it will cause the index to be 0, causing the index inserted into the subscribe_list to be 0, which will cause mcp::subscribe::subscription to return an error.

```
int mcp::subscribe::add(std::string subscribe,int indexno)
{
    int index = is_subscribe_exist(subscribe);
    if (index > 0)//exist
    {
        return index;
    }
    else//insert
    {
        if (indexno < 1)
        {
            index = get_max_index() + 1;
        }
        else
        {
            if (index_is_exist(indexno))
                return -1;
            index = indexno;
        }
        subcribe_list.insert(std::pair<int,            std::string>(index,
subscribe));
    }
    return index;
}


mcp::rpc_ws_error      mcp::subscribe::subscription(std::string        message,
mcp::rpc_ws_connection & conn)
```



```
{
    int index = 0;
    rpc_ws_error ret = rpc_ws_error::success;
    index = get_index_by_message(message);
    if (index == 0)//not exist
    {
        ret = rpc_ws_error::message_not_exist;
        return ret;
    }
```

Result: Confirmed

Fix Result: Fixed

3.8 Nullptr access may lead to Node DoS

ID:	NVE-008	**Location:**	mcp/rpc/rpc_ws.cpp
Severity:	Low	**Category:**	Transaction processing Security
Likelihood:	Low	**Impact:**	Low

Description:

strlen does not determine whether handler->response.c_str() is empty, which will cause null pointer access. It is recommended to use strlen_s.

```
void mcp::rpc_ws_connection::on_read(
    boost::system::error_code ec,
    std::size_t bytes_transferred)
{
    boost::ignore_unused(bytes_transferred);

    // This indicates that the session was closed
    if (ec == boost::beast::websocket::error::closed)
    {
        rpc_ws.close_ws(*this);
        return;
    }

    if (ec)
    {
        LOG(m_log.error) << boost::str(boost::format("Error read data
WebSocket RPC connections: %1%") % ec);
    }

    // deal the message
    auto handler(std::make_shared<mcp::rpc_ws_handler>(this->rpc_ws, *this,
to_string(buffer.data())));
    handler->process_request();

    if (strlen(handler->response.c_str()) > 0)
    {
```

Result: Confirmed

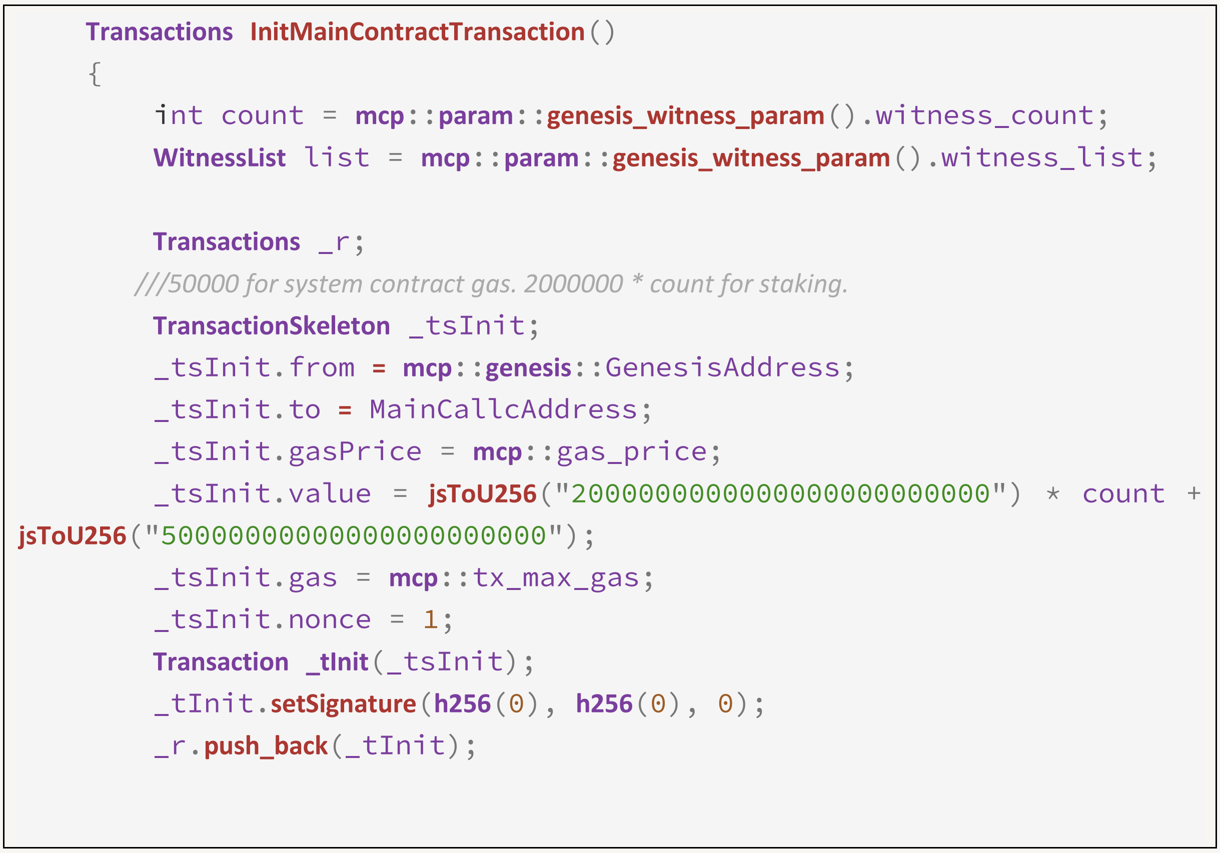
Fix Result: Fixed

3.9 count should not use sign type

ID:	NVE-009	**Location:**	mcp/core/contract.cpp
Severity:	Low	**Category:**	Transaction processing Security
Likelihood:	Low	**Impact:**	Low

Description:

In here count is sign. So if count>0x8000000 will lead of tsInit.value overflow， become a small integer.



```
    Transactions InitMainContractTransaction()
    {
        int count = mcp::param::genesis_witness_param().witness_count;
        WitnessList list = mcp::param::genesis_witness_param().witness_list;

        Transactions _r;
       ///50000 for system contract gas. 2000000 * count for staking.
        TransactionSkeleton _tsInit;
        _tsInit.from = mcp::genesis::GenesisAddress;
        _tsInit.to = MainCallcAddress;
        _tsInit.gasPrice = mcp::gas_price;
        _tsInit.value = jsToU256("2000000000000000000000000") * count +
jsToU256("50000000000000000000000");
        _tsInit.gas = mcp::tx_max_gas;
        _tsInit.nonce = 1;
        Transaction _tInit(_tsInit);
        _tInit.setSignature(h256(0), h256(0), 0);
        _r.push_back(_tInit);
```

Result: Confirmed

Fix Result: Fixed

4. CONCLUSION

In this audit, we thoroughly analyzed **OORT** Blockchain 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-Sec

ExVul