Tonark Smart Contract

SMART CONTRACT AUDIT REPORT

September 2024

ExVul

www.exvul.com

Table of Contents

1. EXECUTIVE SUMMARY

Exvul Web3 Security was engaged by tonark 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.

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
Additional Recommendations	Semantic Consistency Checks
	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

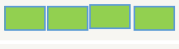
Project Name: tonark

Audit Time: September 20, 2024 – September 30, 2024

Language: FunC

File Name	Link

2.2 Summary

Severity	Found	
Critical	1	
High	0	
Medium	0	
Low	2	
Informational	4	

2.3 Key Findings

ID	Severity	Findings Title	Status	Confirm
NVE-001	**Critical**	3.1 The "remainder" field can only be the last field of the struct	Fixed	Confirmed
NVE-002	**Low**	3.2Outdated compiler version	Fixed	Confirmed
NVE-003	**Low**	3.3 Unused Message	Fixed	Confirmed
NVE-004	**Informational**	3.4 The same constant	Fixed	Confirmed
NVE-005	**Informational**	3.5 Inconsistent Code Standards	Ignored	Confirmed
NVE-006	**Informational**	3.6 The same comment	Fixed	Confirmed
NVE-007	**Informational**	3.7 Unused parameters	Fixed	Confirmed

Table 2.3: Key Audit Findings

3. DETAILED DESCRIPTION OF FINDINGS

3.1 The "remainder" field can only be the last field of the struct

ID:	NVE-001	Location:	
Severity:	Critical	Category:	Business Issues
Likelihood:	Critical	Impact:	Critical

Description:

To prevent misuse of the contract storage and reduce gas consumption, make sure to specify remaining serialization option only on the last field of the given Message.

```
48  message(0x58569ad1) WithdrawInternal {
49      account: Address;
50      token: Address;
51      amount: Int;
52      custom_payload: Cell?;
53      forward_ton_amount: Int as coins;
54      forward_payload: Slice as remaining;
55      tokenLength: Int as uint32;
56      marketInfo: map<Address, MarketInfo>;
57  }
```

```
65  message(0x2c554235) BorrowInternal {
66      account: Address;
67      token: Address;
68      amount: Int;
69      custom_payload: Cell?;
70      forward_ton_amount: Int as coins;
71      forward_payload: Slice as remaining;
72      tokenLength: Int as uint32;
73      marketInfo: map<Address, MarketInfo>;
74  }
```

Result: fixed

3.2 Outdated compiler version

ID:	NVE-002	Location:	
Severity:	Low	Category:	compiler
Likelihood:	High	Impact:	High

Description:

The Tact language is continually being developed and updated, with each update addressing numerous security vulnerabilities. We have detected several behaviors in the code that are prohibited by the latest version of the compiler (1.5.2). It is recommended to update the compiler and other development SDKs to the most recent versions.

Result: fixed

3.3 Unused Message

ID:	NVE-003	Location:	
Severity:	Low	Category:	Business Issues
Likelihood:	High	Impact:	Low

Description:

Unused Messages were found in the code, it can be removed.

```
23
24  message(0x5860390b) TokenInInfo {
25      from: Address;
26      token: Address;
27      amount: Int;
28  }
29
```

```
147
148  message(0xe6a18541) SupplyNotification {
149      account: Address;
150      token: Address;
151      amount: Int;
152  }
153
```

Result: fixed

3.4 The same constant

ID:	NVE-004	Location:	
Severity:	**Informational**	Category:	Business Issues
Likelihood:	**Informational**	Impact:	**Informational**

Description:

There are two constants with the same value here. Optimizing this could reduce gas consumption.

```
278  const SECONDS_PER_YEAR: Int = 31536000;
279  const BASE: Int = pow(10, 9);
280  // power index max (11, 20)
281  const SCALE: Int = pow(10, 9);
282  const SCALE_BASE: Int = pow(10, 18);
```

Result: fixed

3.5 Inconsistent Code Standards

ID:	NVE-005	Location:	
Severity:	**Informational**	Category:	Business Issues
Likelihood:	**Informational**	Impact:	**Informational**

Description:

In the two view functions, marketInfo and tokenInfo, marketInfo provides the raw results without invoking the dp function, whereas tokenInfo presents the processed outcomes after invoking the dp function. The discrepancy between these two could potentially mislead developers.

```
963     get fun marketInfo(): map<Address, MarketInfo> {
964         let infos: map<Address, MarketInfo> = emptyMap();
965         let i: Int = 0;
966         while (i < self.tokenLength) {
967             let token: Address = self.tokenIndex.get(i)!!;
968             let info: Market = self.markets.get(token)!!;
969             infos.set(
970                 token,
971                 MarketInfo {
972                     price: self.tokenPrice.get(token)!!,
973                     supplyIndex: info.supplyIndex,
974                     borrowIndex: info.borrowIndex,
975                     collateralFactor: info.collateralFactor,
976                     valid: info.valid
977                 }
978             );
979             i += 1;
980         }
981         return infos;
982     }
983
984     get fun tokenInfo(token: Address): Market {
985         self.onlyValidToken(token);
986         let info: Market = self.markets.get(token)!!;
987         return Market {
988             totalSupply: self.dp(info.totalSupply),
989             totalBorrows: self.dp(info.totalBorrows),
990             supplyIndex: self.dp(info.supplyIndex),
991             borrowIndex: self.dp(info.borrowIndex),
992             totalReserves: self.dp(info.totalReserves),
993             reserveFactor: self.dp(info.reserveFactor),
994             timestamp: info.timestamp,
995             collateralFactor: self.dp(info.collateralFactor),
996             valid: info.valid
997         };
998     }
999
```

Result: no need to fix

Fix Result: Ignore

customer:

MarketInfo is for use in the contract, so it uses the original data in the contract. TokenInfo is displayed to users, so it uses processed data

3.6 The same comment

ID:	NVE-006	Location:	
Severity:	Informational	Category:	Business Issues
Likelihood:	Informational	Impact:	Informational

Description:

The Market contract and the Ledger contract both utilize the same comment in their sendTon function.

In Market contract:

```
1076
1077    inline fun sendTon(to: Address, value: Int, mode: Int) {
1078        send(SendParameters {
1079            to: to,
1080            value: value,
1081            mode: mode,
1082            bounce: false,
1083            body: "Ark Market Send Back TON".asComment()
1084        });
1085    }
1086
```

In Ledger contract:

```
1465
1466    inline fun sendTon(to: Address, value: Int, mode: Int) {
1467        send(SendParameters {
1468            to: to,
1469            value: value,
1470            mode: mode,
1471            bounce: false,
1472            body: "Ark Market Send Back TON".asComment()
1473        });
1474    }
1475
```

Result: fixed

3.7 Unused parameters

ID:	NVE-007	Location:	
Severity:	Informational	Category:	Business Issues

Likelihood:	**Informational**	**Impact:**	**Informational**

Description:

In the following process:
Borrow -> BorrowInternal -> BorrowNotification
Withdraw-> WithdrawInternal-> WithdrawInternal

Here, there are two parameters, custom_payload and forward_payload, that are controllable by users. However, these parameters do not participate in the execution logic and are solely used as parameters for contract transfers. This could potentially introduce hidden security risks.

Result: Fixed

4. CONCLUSION

In this audit, we thoroughly analyzed **tonark** 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-Sec

ExVul