Tokamak Network - DAO

Security Assessment

2021. 02. 25

Carl Park @4000d

Disclaimer

This report should not be considered to ensure the investment of a particular team or project, or the suitability of a business model.

This report should not be used for decision making to invest or participate in a particular project.

This report identifies vulnerabilities that were overlooked during development and areas requiring additional security measures. However, this does not mean discovering all vulnerabilities and does not guarantee that the source code is completely secure, even if no vulnerabilities are found. This do not provide a complete guarantee of all vulnerabilities and types of attacks that exist even after auditing is complete.

Overview

Project Summary

The audited contract is a DAO for decentralized management of TON ownership. Users can use their TON to create an agenda or vote on an existing agenda. These user actions are made through the Candidate Contract.

Contracts are complied with ERC20 and ERC165 correctly and have well implemented to ensure that the user's actions are ultimately executed through DAO. However, there was a vulnerability for a malicious attacker to obtain the right to vote without depositing TON by misleading the criteria for selecting the winner or loser of the two contestants competing for the right to vote (TO-20).

Audit Summary

Delivery Date: 2021. 02. 25
Timeline: 2021. 02. 01 - 2021. 02. 25

Vulnerability Summary

Total Findings: 25

- informational: 20
- minor: 3
- critical: 1
- medium: 1

Files In Scope

- contracts/dao/Candidate.sol
- contracts/dao/DAOAgendaManager.sol
- contracts/dao/DAOCommittee.sol
- contracts/dao/DAOCommitteeProxy.sol
- contracts/dao/DAOVault2.sol
- contracts/dao/StorageStateCommittee.sol
- contracts/factory/CandidateFactory.sol
- contracts/lib/Agenda.sol

List Of Findings

ID	Severity	Title
TO-1	informational	not inheriting interface
TO-2	informational	use default visibility
TO-3	informational	inexistent input sanitization (non-zero & Address.isContract)
TO-4	informational	comparison with boolean literal
TO-5	informational	useless type cast
TO-6	informational	public function instead of external function
TO-7	informational	use uint instead of uint256
TO-8	informational	inconsistent reason string prefix
TO-9	informational	inefficient function implementation (_getCoinageToken)
TO-10	informational	calculate seconds instead of using `days` keyword
TO-11	minor	inexistent unexpected enum value check
TO-12	informational	redundant require (validAgenda modifier)
TO-13	informational	emit event before state mutation
TO-14	informational	no emitted event
TO-15	minor	invalid opcode
TO-16	informational	inefficient require order
TO-17	informational	empty reason string
TO-18	informational	inefficient state layout
TO-19	informational	require to modifier (heavy reason string size)
TO-20	critical	anyone can control commitee without any cost

TO-21	medium	OOG may prevent an agenda from being executed
TO-22	minor	always return false
TO-24	informational	duplicate public function _implementation() and implementation()
TO-25	informational	disparity of reason string (_impl)

TO-1 - not inheriting interface

ID	Type	Severity	Location
TO-1	Coding Style	informational	dao/Candidate.sol#L18 dao/DAOAgendaManager.sol#L12 dao/DAOCommittee.sol#L17 dao/DAOVault2.sol#L10

These contracts have corresponding interfaces, but they do not inherit their interfaces. It is highly recommended that the interface be inherited to validate it at compile time because it is very important to ensure that the contract has implemented the interface correctly.

Alleviations

fix commit: 572e0ad7499114d778fcf1add81e3f02c17a2de2

TO-2 - use default visibility

ID	Type	Severity	Location
TO-2	Volatile Code	informational	dao/Candidate.sol#L21

The `isLayer2Candidate` variable has an implicit internal visibility because it does not specify visibility. If the Visibility specifier is not specified, there is a possibility that the functionality of that contract and other interactive contacts will not work correctly in the future.

Recommendation

It is recommend to explicitly assign visibility to the `isLayer2Candidate` variable.

Alleviations

fix commit: 572e0ad7499114d778fcf1add81e3f02c17a2de2

TO-3 - inexistent input sanitization (non-zero & Address.isContract)

ID	Type	Severity	Location
TO-3	Volatile Code	informational	dao/Candidate.sol#L39 dao/Candidate.sol#L42-L43 dao/Candidate.sol#L64 dao/Candidate.sol#L70 dao/DAOCommitteeProxy.sol#L21-L22 dao/DAOCommitteeProxy.sol#L23 dao/DAOCommitteeProxy.sol#L24 dao/DAOCommitteeProxy.sol#L25 dao/DAOCommitteeProxy.sol#L26 dao/DAOCommitteeProxy.sol#L28 dao/DAOCommitteeProxy.sol#L55 dao/DAOVault2.sol#L24 dao/DAOVault2.sol#L31 dao/DAOVault2.sol#L37

The function and constructors that take address type parameters do not check whether the values are non-zero. This can lead the contracts to store zero address in their state variables, making a possibility for system malfunction.

Recommendation

It is recommended to determine whether the address type parameters is non-zero.

Alleviations

fix commit: 9e778a8828720f2058d2bb89292f96aed2810d32

TO-4 - comparison with boolean literal

ID	Type	Severity	Location
TO-4	Gas Optimization	informational	dao/Candidate.sol#L78 dao/DAOCommitteeProxy.sol#L70

The expression is unnecessarily compared to the Boolean value. To reduce gas cost, the `==` operator can be omitted.

Recommendation

It is recommended to removi the `==` operator as shown in the source code below.

```
// instead of this
require(isLayer2Candidate == false, "...");

// use this
require(!isLayer2Candidate, "...");
```

Alleviations

fix commit: 879d53efd69232c3dde437b8fc57a79fb39c2cbe

TO-5 - useless type cast

ID	Type	Severity	Location
TO-5	Language Specific	informational	dao/Candidate.sol#L82

`seigManager` variable is already `ISeigManager` type, so no type cast is required.

Alleviations

fix commit: 38a176d374a04950527ecdc7405406f7d40c2e5f

TO-6 - public function instead of external function

ID	Type	Severity	Location
TO-6	Gas Optimization	informational	dao/Candidate.sol#L98 dao/Candidate.sol#L111 dao/Candidate.sol#L119 dao/Candidate.sol#L123-L124 dao/Candidate.sol#L125 dao/DAOAgendaManager.sol#L339 dao/DAOAgendaManager.sol#L343 dao/DAOAgendaManager.sol#L349 dao/DAOAgendaManager.sol#L372 dao/DAOAgendaManager.sol#L389 dao/DAOAgendaManager.sol#L409 dao/DAOAgendaManager.sol#L413 dao/DAOCommittee.sol#L96 dao/DAOCommittee.sol#L108 dao/DAOCommittee.sol#L124 dao/DAOCommittee.sol#L135 dao/DAOCommittee.sol#L142 dao/DAOCommittee.sol#L149 dao/DAOCommittee.sol#L156 dao/DAOVault2.sol#L44 dao/DAOVault2.sol#L52

The public functions above are not used inside the contract. If they are external functions, contract deploy gas cost can be decreased by reducing the contract size.

Recommendation

It is recommended to change the mentioned public functions to external functions.

Alleviations

fix commit: e978a26b54b6879d1ce374ed01d876c66e8d1e60

TO-7 - use uint instead of uint256

ID	Type	Severity	Location
TO-7	Language Specific	informational	dao/Candidate.sol#L108 dao/StorageStateCommittee.sol#L21-L22

Overall, the uint type is used together with uint256 type. uint is the alias of uint256, but it is recommended to use uint256 data type without using the alias to make the code base clean.

(Location list is missing due to the size.)

Alleviations

fix commit: 706fe30a3336d9b92e8ce71af2cec47d13ac2b76

TO-8 - inconsistent reason string prefix

ID	Type	Severity	Location
TO-8	Coding Style	informational	dao/Candidate.sol#L136 dao/Candidate.sol#L151 dao/DAOCommittee.sol#L184 dao/DAOCommittee.sol#L223 dao/DAOCommittee.sol#L308 dao/DAOCommittee.sol#L616-L617

Many of the reason strings use the name of the contract as a prefix throughout the code base. However, the linked require statements use the name of the other contract as a prefix. This makes it difficult to debug and understand the actual behavior of contracts.

Recommendation

It is recommended that the contract name as a prefix of the reason string.

Alleviations

fix commit: 9c36423dbcd1c84c0f1c4f195afd17083d2a669e

TO-9 - inefficient function implementation (_getCoinageToken)

ID	Type	Severity	Location
TO-9	Gas Optimization	informational	dao/Candidate.sol#L136 dao/Candidate.sol#L151 dao/Candidate.sol#L155

The `_getCoinageToken` function is used in the same way as the `totalStaked` function and the `stakedOf` function. Therefore, if the non-zero confirmation and return type are specified as IERC20 in the `_getCoinageToken` function, the implementation of the public functions is simplified and the contract size can be reduced at the same time.

Alleviations

fix commit: 56f1d1a8922aeb4943fad9e2233e0ab91c336128

TO-10 - calculate seconds instead of using `days` keyword

ID	Type	Severity	Location
TO-10	Coding Style	informational	dao/DAOAgendaManager.sol#L46-L47

Numerical operations can be skipped to keep the code clean using `days` keyword that are available globally in solidity. It is recommended to use `days`.

Alleviations

fix commit: 1f4b4746f71e324dbce1cf60bdcb004a219be8e0

TO-11 - inexistent unexpected enum value check

ID	Type	Severity	Location
TO-11	Volatile Code	minor	dao/DAOAgendaManager.sol#L62

`LibAgenda.AgendaStatus` is an enum type with size of 6. If the contract set `LibAgenda.AgendaStatus.None` for `_status` value greater than 6 in `getStatus` function, which is used by the `setStatus` function, it cannot detect an unexpected `_status` value. To prevent this, it is recommended to check that the `_status` parameter is below 6.

Alleviations

fix commit: 2b03094442c5e2dd18a5e564494565e62577815b

TO-12 - redundant require (validAgenda modifier)

ID	Type	Severity	Location
TO-12	Gas Optimization	informational	dao/DAOAgendaManager.sol#L76 dao/DAOAgendaManager.sol#L204 dao/DAOAgendaManager.sol#L295 dao/DAOAgendaManager.sol#L300 dao/DAOAgendaManager.sol#L311 dao/DAOAgendaManager.sol#L316 dao/DAOAgendaManager.sol#L321 dao/DAOAgendaManager.sol#L335 dao/DAOAgendaManager.sol#L344

Above require statements check that the `_agendaID` parameter is valid. However, it is recommended to use one modifier to reduce require statements. In particular, it is recommended to use `validAgenda` modifier to replace the referenced request statement.

Alleviations

fix commit: 23530b23d34d91c2fdcc1e3547acf0b9f97a9fe0

TO-13 - emit event before state mutation

ID	Type	Severity	Location
TO-13	Coding Style	informational	dao/DAOAgendaManager.sol#L79 dao/DAOAgendaManager.sol#L210 dao/DAOAgendaManager.sol#L237 dao/DAOCommittee.sol#L185 dao/DAOCommittee.sol#L330

These codes manipulate the state variables after a event emits. It is recommended to unify the order in which events emit after state variables change throughout the code base.

Alleviations

fix commit: 7fe9b6a6c68ef48a2c63839a75d91bbcd1bb820c

TO-14 - no emitted event

ID	Type	Severity	Location
TO-14	Coding Style	informational	dao/DAOAgendaManager.sol#L85 dao/DAOAgendaManager.sol#L91 dao/DAOAgendaManager.sol#L97 dao/DAOAgendaManager.sol#L141 dao/DAOAgendaManager.sol#L195 dao/DAOCommittee.sol#L171

These functions change important state variables in the system, but do not emit events. It is recommended to emit events for future contract maintenance.

Alleviations

fix commit: ba078835eb7d70a7e91e751eb155c4bb5b09272f

TO-15 - invalid opcode

ID	Type	Severity	Location
TO-15	Coding Style	minor	dao/DAOAgendaManager.sol#L167 dao/DAOAgendaManager.sol#L222 dao/DAOAgendaManager.sol#L235 dao/DAOAgendaManager.sol#L247 dao/DAOAgendaManager.sol#L264 dao/DAOAgendaManager.sol#L281 dao/DAOAgendaManager.sol#L326 dao/DAOAgendaManager.sol#L348 dao/DAOAgendaManager.sol#L364 dao/DAOAgendaManager.sol#L380 dao/DAOAgendaManager.sol#L399 dao/DAOAgendaManager.sol#L414 dao/DAOCommittee.sol#L346

When accessing a state variable whose type is array, if an index that exceeds the length of the array is used, the transaction is reverted due to "invalid opcode". This is a major factor that makes it difficult to identify the cause of the transaction reversion. To prevent this, it is strongly recommended to use `validAgenda` modifier to check the length of the array.

Alleviations

fix commit: 801c0fbb51bcaddaef4268389b9774bb40bbb2a0

TO-16 - inefficient require order

ID	Type	Severity	Location
TO-16	Gas Optimization	informational	dao/DAOAgendaManager.sol#L176 dao/DAOCommittee.sol#L221

Those require statements run after reading other state variables (`isVoter`, `hasVoted`) or interacting with other contract (`layer2Registry.registerAndDeployCoinage`). If they run before reading or interacting, it can reduce the gas cost if a transaction is reverted.

Alleviations

fix commit: 6dadf86731025a199a6067c3c76a8c179ce24f25

TO-17 - empty reason string

ID	Type	Severity	Location
TO-17	Coding Style	informational	dao/DAOAgendaManager.sol#L193

Reason string is not given for the parameter of revert function. If a transaction that execute linked function is reverted, this makes it difficult to identify the cause of the transaction reversion.

Recommendation

Use proper reason string as a parameter of revert function.

Alleviations

fix commit: 299ce749bc867be458f1471b5d5c1cf2f88fb17f

TO-18 - inefficient state layout

ID	Type	Severity	Location
TO-18	Gas Optimization	informational	dao/DAOCommittee.sol#L25-L26 dao/StorageStateCommittee.sol#L21-L22 dao/StorageStateCommittee.sol#L23 dao/StorageStateCommittee.sol#L24

Some of uin256 type members, declared in the `CandidateInfo` and `AgendaCreatingData` struct, store timestamps. Because timestamp does not use more than 128 bits, uint128 can be used to reduce the storage slots occupied by the structure. Therefore, it is recommended to reduce gas cost by replacing the uint256 type members for timestamps with uint128 type.

Alleviations

fix commit: 7a2160601ae70c9df4db7d1ba1f6ba4d42b1fae4

TO-19 - require to modifier (heavy reason string size)

ID	Type	Severity	Location
TO-19	Gas Optimization	informational	dao/DAOCommittee.sol#L97 dao/DAOCommittee.sol#L111 dao/DAOCommittee.sol#L127 dao/DAOCommittee.sol#L136 dao/DAOCommittee.sol#L143 dao/DAOCommittee.sol#L150 dao/DAOCommittee.sol#L157 dao/DAOCommittee.sol#L164

The same require statement is used to determine whether the given address type parameter is non-zero. Also, the contract size is unnecessarily large because string literals with the same meaning are used a lot.

Recommendation

The gas cost can greatly reduce by replacing those require statements with one modifier.

Alleviations

fix commit: 87d07600908e47000d55978d623ea161405ce951

TO-20 - anyone can control commitee without any cost

ID	Type	Severity	Location
TO-20	Control Flow	critical	dao/DAOCommittee.sol#L307

The `changeMember` function make `msg.sender` to obtain voting rights by replacing existing members based on the token the sender deposited. However, the criteria for "how many tokens have been deposited" refer to two participants (`msg.sender`, `prevMemberContract`) competing for membership, rather than to a trusted contract. An attacker can exercise all possible voting rights without depositing tokens by distributing his own `Candidate` Contract as follows:

```
contract CandidateAttacker {
  ... // same implementation as Candidate except `totalStaked` function
  function totalStaked public pure returns (uint256) {
    return 2 ** 256 - 1;
  }
}
```

Recommendation

The amount of tokens deposited by a particular user must be read directly by the `DAOCommitee` Contract through `SeigManager`.

Alleviations

fix commit: cb9128fd48d6d9e7ea99d4496a9bb6a38cfad259, 30d9eb2214d7b4c2f6aa0cb4976ddab2fa92308b

The development team modified to prevent any other untrusted contacts from calling the `changeMember` function by making only `Candidate` contracts created by `createCandidate` function can call `changeMember` function.

TO-21 - OOG may prevent an agenda from being executed

ID	Type	Severity	Location
TO-21	Volatile Code	medium	dao/DAOCommittee.sol#L499

The executable agenda can generate multiple message-calls. If only one message-call fails, the entire message-call might fail. Also, if out of gas happens by the `AgendaExecutionInfo.targets[i]` contract, or if the `AgendaExecutionInfo.targets[i]` contract always reverts the message call (Denial of Service), all messages-calls from that agenda cannot be executed. To prevent this, it is strongly recommended to add a function to recall failed message-calls in the future.

Recommendation

The development team's intention was to implement to call all-or-nothing execution of multiple message-call. I recommend adding `atomic` flags to the `LibAgenda.AgendaExecutionInfo` structure to address existing requirements and recommendations for these vulnerabilities at the same time.

Alleviations

fix commit: 5c33a76e6bde7b288b789e82c768b74ac123c11f

TO-22 - always return false

ID	Type	Severity	Location
TO-22	Volatile Code	minor	dao/DAOCommittee.sol#L526

The `updateSeigniorages` function always returns `false`. Other contracts that depend on the return value of that function can function incorrectly.

Recommendation

It is recommended to modify the function to return `true` if it is terminated successfully.

Alleviations

fix commit: bd02d1a919e8fdc22a621d6bcd0419463c738e3f

TO-24 - duplicate public function _implementation() and implementation()

ID	Type	Severity	Location
TO-24	Gas Optimization	informational	dao/DAOCommitteeProxy.sol#L10 dao/DAOCommitteeProxy.sol#L60

Solidity makes `_implementation` public function because `_implementation` state variable has public visibility. This overlaps with the `implementation` function that performs the same function. Because duplicate public functions increases gas cost, I recommend using only one of them.

Alleviations

fix commit: f7c5807839d10b7b45f2a2e34295ba34d2bde562

TO-25 - disparity of reason string (_impl)

ID	Type	Severity	Location
TO-25	Coding Style	informational	dao/DAOCommitteeProxy.sol#L55

The require statement checks that the state variable and the function parameter are not the same. A given reason string means whether the given parameter is zero or non-zero, which does not match the actual boolean expression.

Alleviations

fix commit: 9342631d90a3764316df81208311bb38a1c4218d

Finding Categories

Gas Optimization

Although it behaves logically the same as existing source code, EVM opcodes can be optimized to reduce transaction fees when deploying or executing a contract.

Control Flow

Consider whether it is correct for the contract to limit its function. This includes features that can only be performed by a specific user, such as an owner of the contract.

Volatile Code

Consider a particular edge case where a particular code behaves unexpectedly or a vulnerability exists.

Language Specific

Consider issues that occur only in Solidity.

Coding Style

The generated byte-code is not affected, but consider modifying the codebase to allow easily maintenance.