decurity

Smart Contract Security Audit Report

Truflation

1. Contents

2. General Information

This report contains information about the results of the security audit of the Truflation (hereafter referred to as "Customer") smart contracts, conducted by Decurity in the period from 10/10/2024 to 18/10/2024.

2.1. Introduction

Tasks solved during the work are:

- Review the protocol design and the usage of 3rd party dependencies,
- Audit the contracts implementation,
- Develop the recommendations and suggestions to improve the security of the contracts.

2.2. Scope of Work

The audit scope included the contracts in the following repository: truflation/truflation-contracts/pull/98. Initial review was done for the commit 23adfc and the re-testing was done for the commit 59a4a1.

The following contracts have been tested:

- src/token/TrufMigrator.sol
- src/token/TrufVesting.sol
- src/token/TruflationToken.sol
- src/token/VotingEscrowTruf.sol

2.3. Threat Model

The assessment presumes actions of an intruder who might have capabilities of any role (an external user, token owner, token service owner, a contract). The centralization risks have not been considered upon the request of the Customer.

The main possible threat actors are:

- User,
- Protocol owner,
- Liquidity Token owner/contract.

The table below contains sample attacks that malicious attackers might carry out.

Table. Theoretically possible attacks

Attack	Actor
Contract code or data hijacking *Deploying a malicious contract or submitting malicious data*	Contract owner Token owner
Financial fraud *A malicious manipulation of the business logic and balances, such as a re-entrancy attack or a flash loan attack*	Anyone
Attacks on implementation *Exploiting the weaknesses in the compiler or the runtime of the smart contracts*	Anyone

2.4. Weakness Scoring

An expert evaluation scores the findings in this report, an impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2.5. Disclaimer

Due to the intrinsic nature of the software and vulnerabilities and the changing threat landscape, it cannot be generally guaranteed that a certain security property of a program holds.

Therefore, this report is provided “as is” and is not a guarantee that the analyzed system does not contain any other security weaknesses or vulnerabilities. Furthermore, this report is not an endorsement of the Customer’s project, nor is it an investment advice.

That being said, Decurity exercises best effort to perform their contractual obligations and follow the industry methodologies to discover as many weaknesses as possible and maximize the audit coverage using the limited resources.

3. Summary

As a result of this work, we have discovered a single critical exploitable security issue along with high, medium issues.

The other suggestions included fixing the low-risk issues and some best practices (see Security Process Improvement).

These vulnerabilities have been addressed and thoroughly re-tested as part of our process.

3.1. Suggestions

The table below contains the discovered issues, their risk level, and their status as of October 24, 2024.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
Categories migration logic is broken	src/token/TrufVesting.sol	**High**	Fixed
Incorrect stake receiver	src/token/TrufVesting.sol	**High**	Fixed
Blacklisted addresses not removed automatically	script/redeploy/02_generate_tree.js	**Medium**	Acknowledged
Last emission may not equal to `maxAllocation`	src/token/TrufVesting.sol	**Low**	Fixed
UniV3 Liquidity providers are unable to migrate	script/redeploy/01_fetch_data.js	**Low**	Acknowledged
Ownable2Step should be used	src/token/TrufMigrator.sol	**Low**	Fixed

Issue	Contract	Risk Level	Status
Dead code	src/token/TrufMigrator.sol src/token/TrufVesting.sol src/token/VotingEscrowTruf.sol	Low	Fixed
Admin can abuse votes in voting escrow	token/TrufVesting.sol	Low	Acknowledged

4. General Recommendations

This section contains general recommendations on how to improve overall security level.

The Findings section contains technical recommendations for each discovered issue.

4.1. Security Process Improvement

The following is a brief long-term action plan to mitigate further weaknesses and bring the product security to a higher level:

- Keep the whitepaper and documentation updated to make it consistent with the implementation and the intended use cases of the system,
- Perform regular audits for all the new contracts and updates,
- Ensure the secure off-chain storage and processing of the credentials (e.g. the privileged private keys),
- Launch a public bug bounty campaign for the contracts.

5. Findings

5.1. Categories migration logic is broken

Risk Level: **High**

Status: Partially fixed in the commit 305b21. Finally fixed in the commit 59a4a1.

Contracts:

- src/token/TrufVesting.sol

Description:

During migration, those assets that have already been claimed by users should be taken into account.

When initializing a category, the transferred amount should be `maxAllocation - totalClaimed`, instead of transferring the entire `maxAllocation` for the selected category ID.

In the current implementation the `totalClaimed` is not been migrated but the `allocated` does, which may create the confusions into the statistic about the category and break the logic of the migration cause extra funds would be transfered.

Transferring all the funds to the vesting contract disrupts the migration logic because funds that should remain on the TrufMigrator contract are being transferred to the vesting contract, covering rewards that have already been claimed.

Remediation:

Consider adding extra logic to the TrufVesting contract to support migrating categories that have been initialized and partially claimed.

5.2. Incorrect stake receiver

Risk Level: **High**

Status: Fixed in the commit 640bb2.

Contracts:

- src/token/TrufVesting.sol

Description:

The vesting migration process calls the `_stake()` function when the locked amount is greater than zero:

```
        if (locked > 0) {
            _stake(msg.sender, categoryId, vestingId, locked,
stakingStartTime, stakingDuration);
        }
```

In this case, the first argument of the `_stake()` function represents the user. However, the `migrate()` function is passing `msg.sender` (the trufMigrator) as the user, but it should be passing the user from the migration function's parameters instead.

Remediation:

Consider updating the parameter from `msg.sender` to `user`.

5.3. Blacklisted addresses not removed automatically

Risk Level: Medium

Status: There is a comment in the script that data modification will have to be done manually after the snapshot is taken, this is done to keep the maximum claimable TRUF to the same max supply as before, and also for example to move the hacked funds back to the wallet they came from.

Contracts:

- script/redeploy/02_generate_tree.js

Description:

The `01_fetch_data.js` script collects the data from the blockchain to prepare the migration of the holders of TRUF token.

There is blacklist array in it:

```
const BLACKLIST = [
    '0xb1CF7880351E6D16313C03a6686B4c8a5bA6372a', // Hacker
    '0xBc1B0a3D3dCf3BbCeEe12ea74fe077817205546F', // Hacker
    '0x35Eea93759CFfA5A6194F8c7C11a3bB62b619605', // Hacker
    '0x1189905020f68304CF1B6B83bC7f8Cc6bB80d514', // Hacker
    '0x7Ae828001eC0870E4A44c11C31AEbBb572b9e4Ea', // Hacker
    '0x671b8e61c5588F6ec78b9cD75511614945bb2968', // Uni V3
    '0x02C4099F663764a8a24F81E0C01e7646bF367b5e', // Arrakis vault, not sure
why there's so much funds still inside
    '0xb854536206EB6C1013b1642b576196E5EF19D7BA', // CCIP LockReleaseTokenPool
    '0xdCfF65e13aeeB5cc250dA25933a5A95b0d16Fe1c', // Virtual staking rewards
    '0xdB4fAcA2582C47D4C8b7aa65645250eDF1A951fD', // Voting escrow truf
    '0xe4a8C47e675C974Ceed9A2C06C0BE7c026fF8D4e', // Vesting
].map((v) => ethers.getAddress(v));
```

This script aggregates data into a JSON file, and if an address is in the blacklist, it flags the address as blacklisted.

However, both the old 02_generate_signatures.js and the new 02_generate_tree.js scripts do not validate the blacklist. This allows hacker to potentially migrate tokens.

Remediation:

Consider excluding blacklisted addresses from the Merkle tree root to prevent them from participating in the migration.

5.4. Last emission may not equal to `maxAllocation`

Risk Level: Low

Status: Fixed in the commit aa4636.

Contracts:

- src/token/TrufVesting.sol

Description:

It's possible to override a category if it hasn't been initialized. One of the parameters that can be overridden is the `maxAllocation`.

This parameter is used when setting the emission for a specific category:

```
        if (emissions.length == 0 || emissions[emissions.length - 1] !=
maxAllocation) {
            revert InvalidEmissions();
        }
```

Overriding the `maxAllocation` parameter could break the emission logic if the schedule has already been set.

Remediation:

Consider deleting the `emissionSchedule` if the `maxAllocation` overriding of the category is being overridden.

5.5. UniV3 Liquidity providers are unable to migrate

Risk Level: Low

Status: Acknowledged

Contracts:

- script/redeploy/01_fetch_data.js

Description:

The data fetch script does not account for liquidity providers' position balances, preventing them from migration to new token version.

Remediation:

Consider adding logic to account for Uniswap liquidity positions in the migration script.

5.6. Ownable2Step should be used

Risk Level: Low

Status: Fixed in the commit 664d51.

Contracts:

- src/token/TrufMigrator.sol

Description:

The TrufMigrator contract is inherited from Ownable contract. The owner may accidentally specify a non-active address and lose access. `Ownable2Step.sol` is more secure due to a 2-stage ownership transfer.

```
contract TrufMigrator {
        ...

    constructor(address _signer) {
        signer = _signer;
        deployer = msg.sender;
    }
    function initialize(IERC20 _trufToken, TrufVesting _trufVesting,
VotingEscrowTruf _veTRUF) external {
        if (msg.sender != deployer) revert(); // @audit Ownable2Step
        ...
```

Remediation:

We recommend using the `Ownable2Step` contract from Open Zeppelin ([Ownable2Step.sol]) instead.

5.7. Dead code

Risk Level: Low

Status: Fixed

Contracts:

- src/token/TrufMigrator.sol
- src/token/TrufVesting.sol
- src/token/VotingEscrowTruf.sol

Description:

In the `TrufMigrator` contract, there is no function to add wallets to the `isBlacklisted` mapping

In the `TrufVesting` contract, the `isVestingMigrated` mapping and the `signer` variable are unused.

In the `VotingEscrowTruf` contract, the `InvalidAmount` and `MaxPointsExceeded` errors are not used.

Remediation:

Consider removing or implementing the necessary logic for the unused mappings, variables, and errors in the respective contracts to avoid dead code and improve contract efficiency.

5.8. Admin can abuse votes in voting escrow

Risk Level: Low

Status: Acknowledged

Contracts:

- token/TrufVesting.sol

Location: Function: `cancelVesting`.

Description:

`_unstake` function in `VotingEscrowTruf` contract has force parameter. This parameter allows to end vesting before the end time. This can be abused by the admin of the `TrufVesting` contract in the following way:

- Create vesting for any address for a very short period
- Invoke `stake` on `TrufVesting` for max duration possible. This will result in points that are obtained from `VotingEscrowTruf` being maximised.
- Use obtained vote tokens to vote in DAO/etc
- Invoke `cancelVesting` on `TrufVesting` and get tokens released back.

This results in the fact that a privileged role may manipulate voting results without a need for their tokens to actually be locked. They can also gain max voting power from their tokens because the maximum duration can be passed.

Remediation:

Consider moving `cancelVesting` to DAO, so it can be done only in urgent cases.

6. Appendix

6.1. About us

The Decurity team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we've gained expertise in the blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.