CERTIK

Preliminary Comments

Human Protocol

May 3rd, 2022

Table of Contents

Summary

This report has been prepared for Human Protocol to discover issues and vulnerabilities in the source code of the Human Protocol project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Human Protocol
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/humanprotocol/hmt-escrow/
Commit	8d2a8d77383c70e2bc0221a6cfdf908d465da552 a4a792c0c80a238bf66cfc88044bd614e4b0280f

Audit Summary

Delivery Date	May 03, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

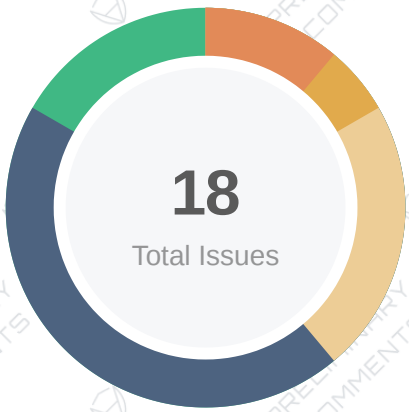
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	2	2	0	0	0	0	0
Medium	1	1	0	0	0	0	0
Minor	4	4	0	0	0	0	0
Informational	8	8	0	0	0	0	0
Discussion	3	3	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
HMT	HMToken.sol	492a1397441cc2a66fd67ffd77bc72447035d14f74b94eed98ff4ca749a04110
ESR	Escrow.sol	7190f1fbdaa2f1d225b1c5214fef03187d36c5120abca1cc5fd5cdcaec6d5256
SMU	SafeMath.sol	5728de7281ca42f0154e39897c0ff9899e13beed36ae6028f0a5dab1b72ae493
SMB	SafeMath.sol	5728de7281ca42f0154e39897c0ff9899e13beed36ae6028f0a5dab1b72ae493
ESC	Escrow.sol	a64bb4080bab83f3cfac6da20795f6eb16841b0b05ef42963147631d2bbb51dd
EFU	EscrowFactory.sol	aa1ac3e2946f66079ec5a4f2091cefe0d2dd16cba5a5b8cb691f7ef6f59cd52f
HMI	HMTokenInterface.sol	e9d71660508b065f865cc54bd5acec8bcdf98b8ecb2a381ad53d1e1a1dc29c92
HMO	HMToken.sol	492a1397441cc2a66fd67ffd77bc72447035d14f74b94eed98ff4ca749a04110
EFB	EscrowFactory.sol	aa1ac3e2946f66079ec5a4f2091cefe0d2dd16cba5a5b8cb691f7ef6f59cd52f
HTI	HMTokenInterface.sol	e9d71660508b065f865cc54bd5acec8bcdf98b8ecb2a381ad53d1e1a1dc29c92

Findings



Critical	0 (0.00%)
Major	2 (11.11%)
Medium	1 (5.56%)
Minor	4 (22.22%)
Informational	8 (44.44%)
Discussion	3 (16.67%)

ID	Title	Category	Severity	Status
ESC-01	Potential Logical Issue In `bulkPayOut()`	Gas Optimization, Logical Issue	Minor	Pending
ESC-02	External Call Inside Loop	Control Flow	Informational	Pending
HMT-01	Initial Token Distribution	**Centralization / Privilege**	**Major**	Pending
HMT-02	Centralization Risk In `Escrow.sol`	**Centralization / Privilege**	**Major**	Pending
HMT-03	Check Effect Interaction Pattern Violated	Volatile Code	Medium	Pending
HMT-04	Misleading Function Name	Coding Style	Minor	Pending
HMT-05	Missing Zero Address Validation	Volatile Code	Minor	Pending
HMT-06	Function 'transferFrom' Of Contract HMToken Does Not Prevent Transfers From The Zero Address	Logical Issue	Minor	Pending
HMT-07	Unoptimized Code	Coding Style, Gas Optimization	Informational	Pending
HMT-08	Unnecessary Use Of `SafeMath` Functions	Gas Optimization, Mathematical Operations	Informational	Pending
HMT-09	Redundant Check	Volatile Code, Gas Optimization	Informational	Pending

ID	Title	Category	Severity	Status
HMT-10	Improper Usage Of `public` And `external` Type	Gas Optimization	Informational	Pending
HMT-11	Missing Emit Events	Coding Style	Informational	Pending
HMT-12	Too Many Digits	Coding Style	Informational	Pending
HMT-13	`bulkPayOut()` Can Still Be Called When Status Is `Complete` Or `Cancelled`	Logical Issue, Control Flow	Informational	Pending
HMT-14	Logic Issue In Function `increaseApproval()`	Logical Issue, Gas Optimization	Discussion	Pending
HMT-15	Inconsistency Between Comment And Code	Inconsistency	Discussion	Pending
HMT-16	Unclear Purpose Of `storeResults()`	Gas Optimization	Discussion	Pending

ESC-01 | Potential Logical Issue In `bulkPayOut()`

Category	Severity	Location	Status
Gas Optimization, Logical Issue	Minor	Escrow.sol (a4a792c): 156~160	Pending

Description

In the function `bulkPayOut()`, in the following piece of code:

```
157         for (uint256 i; i < _amounts.length; i++) {
158             aggregatedBulkAmount += _amounts[i];
159         }
160         require(aggregatedBulkAmount < BULK_MAX_VALUE, "Bulk value too high");
```

it is possible to bypass the check if the sum of the `_amounts[i]` is greater than `max(uint256)`.

Recommendation

We advise checking that no overflow happens to prevent useless computation and unexpected behavior.

ESC-02 | External Call Inside Loop

Category	Severity	Location	Status
Control Flow	Informational	Escrow.sol (a4a792c): 177~179	Pending

Description

External calls are made inside a *for* loop. This might lead to a denial-of-service attack. If any of the calls fail, it will cause the entire loop to revert.

```
177         for (uint i = 0; i < _recipients.length; ++i) {
178             token.transfer(_recipients[i], finalAmounts[i]);
179         }
```

Recommendation

We recommend using the pull-over-push strategy for external calls.

HMT-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	HMToken.sol (a4a792c): 39; HMToken.sol (8d2a8d7): 39	Pending

Description

All of the `HMToken` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute `HMToken` tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

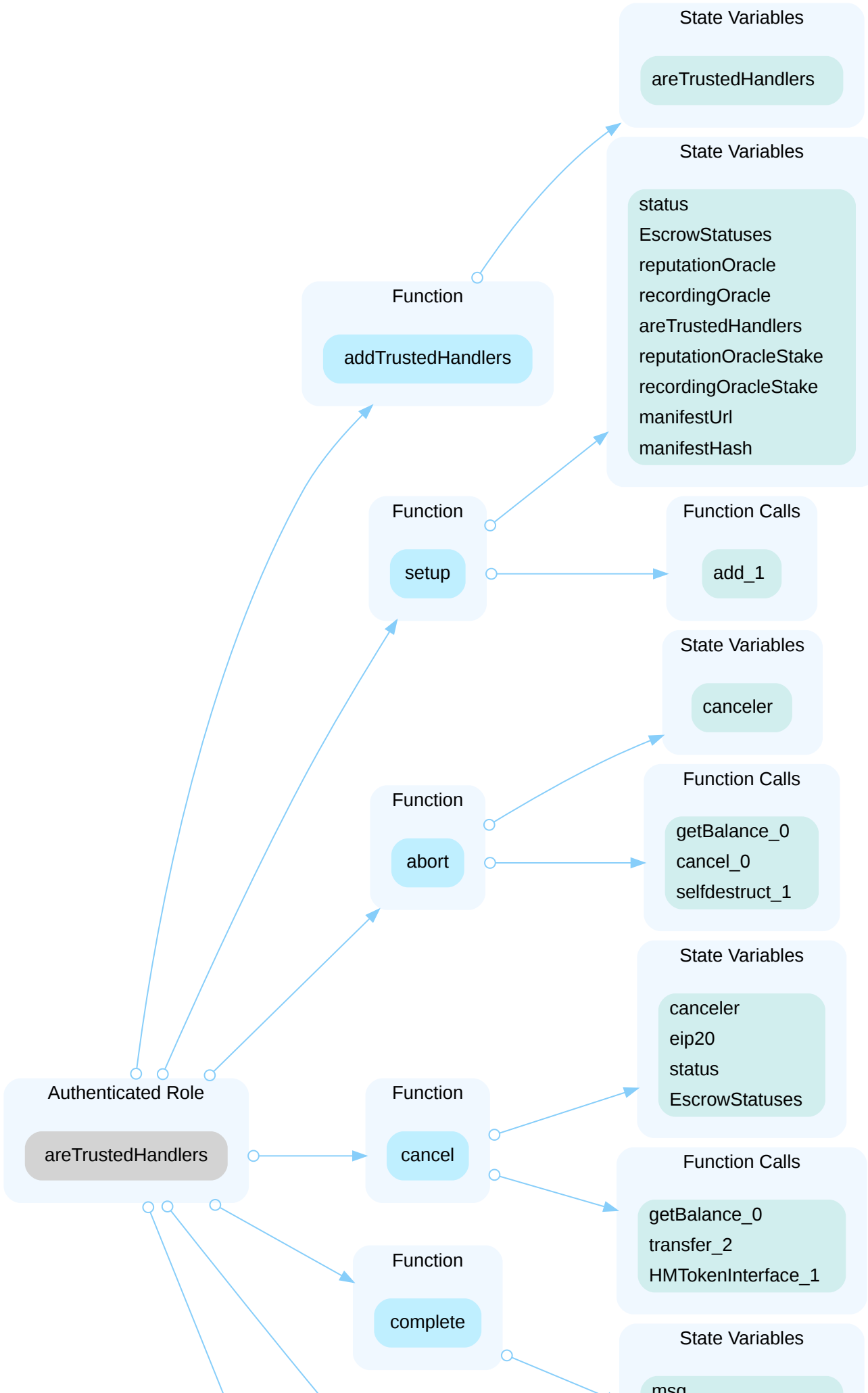
HMT-02 | Centralization Risk In `Escrow.sol`

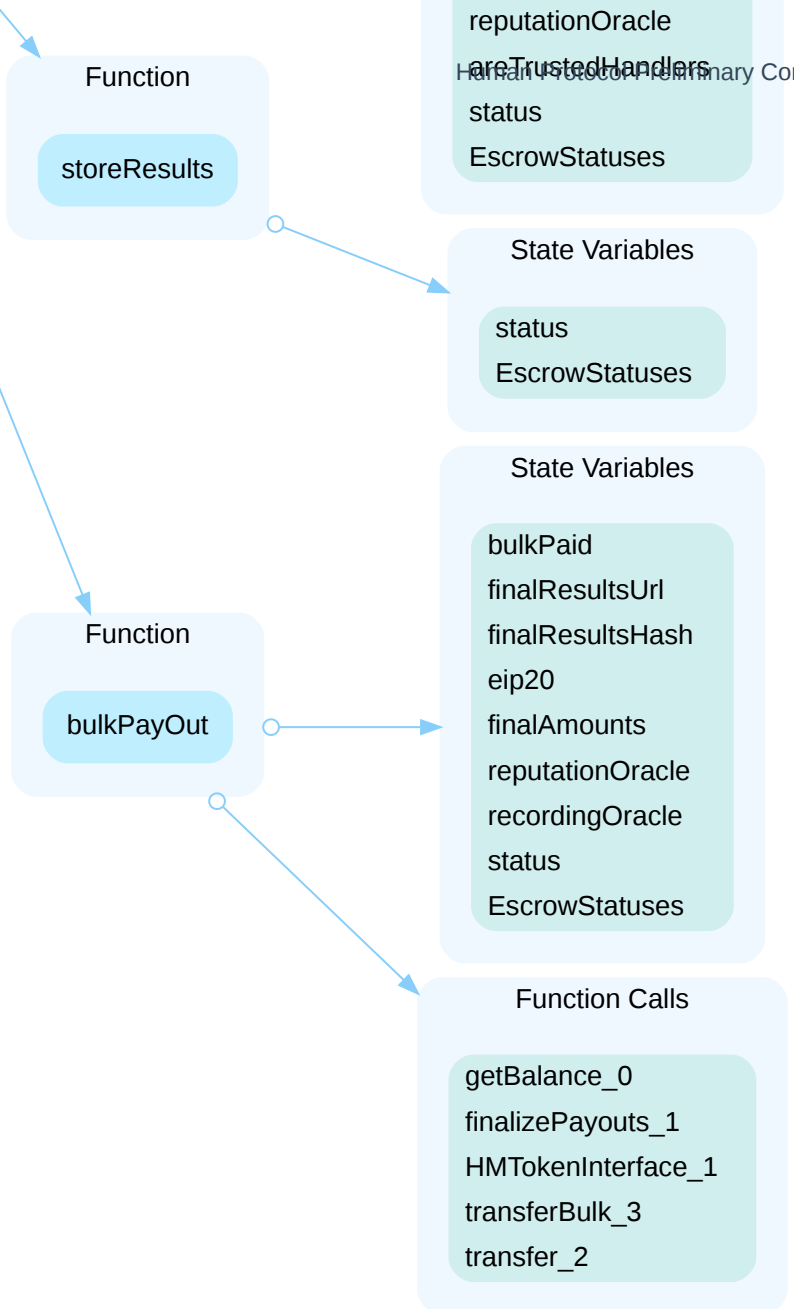
Category	Severity	Location	Status
Centralization / Privilege	**Major**	Escrow.sol (a4a792c): 61~66, 71~110, 112~117, 119~123, 125~132, 134~141, 143~195; Escrow.sol (8d2a8d7): 58~63, 68~107, 109~114, 116~120, 122~129, 131~138, 140~186	Pending

Description

In the contract `Escrow` the role `areTrustedHandlers` has authority over the functions shown in the diagram below.

Any compromise to a trusted handler account may allow the hacker to take advantage of this authority and for example: set all the addresses he/she wants as a trusted handler, because of the logic of the contract these privileges are not revocable; then a lot of addresses will have the power to call functions that should be protected, this could sabotage the contract.


State Variables
areTrustedHandlers
State Variables
status
EscrowStatuses
reputationOracle
recordingOracle
areTrustedHandlers
reputationOracleStake
recordingOracleStake
manifestUrl
manifestHash
Function
addTrustedHandlers
Function
setup
Function Calls
add_1
State Variables
canceler
Function
abort
Function Calls
getBalance_0
cancel_0
selfdestruct_1
State Variables
canceler
eip20
status
EscrowStatuses
Authenticated Role
areTrustedHandlers
Function
cancel
Function Calls
getBalance_0
transfer_2
HMTokenInterface_1
Function
complete
State Variables
msg



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

HMT-03 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Volatile Code	Medium	Escrow.sol (a4a792c): 119~123, 143~189; Escrow.sol (8d2a8d7): 116~120, 140~186	Pending

Description

A reentrancy attack can occur when the contract creates a function that makes an external call to another untrusted contract before resolving any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the external call resolved the effects.

Recommendation

We recommend using the Checks-Effects-Interactions Pattern to avoid the risk of calling unknown contracts or applying OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier for the aforementioned functions to prevent reentrancy attack.

HMT-04 | Misleading Function Name

Category	Severity	Location	Status
Coding Style	Minor	HMToken.sol (a4a792c): 127, 139; HMToken.sol (8d2a8d7): 127, 139	Pending

Description

The function `approveBulk()` does not allow a user to approve certain addresses to transfer certain values, contrary to what the name suggests, but it allows the user to increase the values already approved for this address.

In particular, a user could believe he/she can reduce the allowances of a list of addresses, however, it is only possible to increase the allowances.

Recommendation

We advise renaming this function `increaseApprovalBulk()` to avoid any mistakes.

HMT-05 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	Escrow.sol (a4a792c): 47, 51; EscrowFactory.sol (a4a792c): 17; Escrow.sol (8d2a8d7): 44, 48; EscrowFactory.sol (8d2a8d7): 17	Pending

Description

Addresses should be checked before assignment or external calls to make sure they are not zero addresses.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

HMT-06 | Function 'transferFrom' Of Contract HMToken Does Not Prevent Transfers From The Zero Address

Category	Severity	Location	Status
Logical Issue	Minor	HMToken.sol (8d2a8d7): 48~62; HMToken.sol (a4a792c): 48~62	Pending

Description

It is expected that calls of the form `transferFrom(from, dest, amount)` fail if the address value provided in the `from` in-parameter is the zero address.

Recommendation

Ensure that all calls of the form `transferFrom(from, dest, amount)` in contract `HMToken` revert if invoked with the zero address in its `from` parameter.

HMT-07 | Unoptimized Code

Category	Severity	Location	Status
Coding Style, Gas Optimization	Informational	HMToken.sol (a4a792c): 150~152; HMToken.sol (8d2a8d7): 150~152	Pending

Description

In the function `transferQuiet()`, the following block:

```
150         if (_to == address(0)) return false; // Preclude burning tokens to
uninitialized address.
151         if (_to == address(this)) return false; // Preclude sending tokens to the
contract.
152         if (balances[msg.sender] < _value) return false;
```

is equivalent to :

```
        if ((_to == address(0)) || (_to == address(this)) || (balances[msg.sender] <
_value)) return false;
```

Recommendation

We advise rewriting the code as above to optimize gas consumption and code legibility.

HMT-08 | Unnecessary Use Of `SafeMath` Functions

Category	Severity	Location	Status
Gas Optimization, Mathematical Operations	Informational	HMToken.sol (a4a792c): 80, 83, 93, 96; HMToken.sol (8d2a8d7): 80, 83, 93, 96	Pending

Description

In the linked code, arithmetical operations are performed after a conditional statement has occurred preventing any risk of integer underflow/overflow.

Recommendation

We advise using the classic arithmetic operator when there is no risk of integer overflow/underflow.

HMT-09 | Redundant Check

Category	Severity	Location	Status
Volatile Code, Gas Optimization	Informational	Escrow.sol (a4a792c): 88~92, 126~129; Escrow.sol (8d2a8d7): 85~89, 123~126	Pending

Description

In the function `setup()`, in the following piece of code:

```
85          uint256 totalStake = _reputationOracleStake.add(_recordingOracleStake);
86          require(
87              totalStake >= 0 && totalStake <= 100,
88              "Stake out of bounds"
89          );
```

`totalStake` is an `uint256`, hence it cannot be less than zero.

In the function `complete()`, in the following check:

```
123         require(
124             msg.sender == reputationOracle || areTrustedHandlers[msg.sender],
125             "Address calling is not trusted"
126         );
```

the `reputationOracle` is always a trusted handler, hence requiring `areTrustedHandlers[msg.sender]` is enough.

Recommendation

We advise removing the redundant code.

HMT-10 | Improper Usage Of `public` And `external` Type

Category	Severity	Location	Status
Gas Optimization	Informational	HMToken.sol (a4a792c): 42, 48, 68, 89; Escrow.sol (a4a792c): 112, 125; HMToken.sol (8d2a8d7): 42, 48, 68, 89; Escrow.sol (8d2a8d7): 109, 122	Pending

Description

`public` functions that are never called by the contract could be declared as `external`. `external` functions are more efficient than `public` functions.

Recommendation

Consider using the external attribute for public functions that are never called within the contract.

HMT-11 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	Escrow.sol (a4a792c): 61~66, 112~117, 119~123, 125~132; Escrow.sol (8d2a8d7): 58~63, 109~114, 116~120, 122~129, 140~186	Pending

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

HMT-12 | Too Many Digits

Category	Severity	Location	Status
Coding Style	Informational	HMToken.sol (a4a792c): 21; HMToken.sol (8d2a8d7): 21; Escrow.sol (a4a792c): 23	Pending

Description

Literals with many digits are difficult to read and review.

Recommendation

We advise the client to use the scientific notation to improve readability, for example:

```
uint256 private constant BULK_MAX_VALUE = 1e9 * (10 ** 18);
```

HMT-13 | `bulkPayOut()` Can Still Be Called When Status Is `Complete` Or `Cancelled`

Category	Severity	Location	Status
Logical Issue, Control Flow	Informational	Escrow.sol (a4a792c): 143~195; Escrow.sol (8d2a8d7): 140~186	Pending

Description

Technically it is still possible to call `bulkPayOut()` even if the current status is `Complete` or `Cancelled` because the function does not prevent the execution with these statuses.
This could lead to potential errors from a user.

Recommendation

We advise adding checks to prevent the possibility to call `bulkPayOut()` when the status is `Cancelled` or `Complete`.

HMT-14 | Logic Issue In Function `increaseApproval()`

Category	Severity	Location	Status
Logical Issue, Gas Optimization	Discussion	HMToken.sol (a4a792c): 80; HMToken.sol (8d2a8d7): 80	Pending

Description

In the function `increaseApproval()`, the following conditionnal statement:

```
  81          if (_oldValue.add(_delta) < _oldValue || _oldValue.add(_delta) >=
MAX_UINT256) { // Truncate upon overflow.
  82              allowed[msg.sender][_spender] = MAX_UINT256.sub(1);
```

is equivalent to:

```
        if (_oldValue.add(_delta) = MAX_UINT256) {
            allowed[msg.sender][_spender] = MAX_UINT256.sub(1);
```

because nothing is greater than `MAX_UINT256` and `add()` from `SafeMath` will revert if `_oldValue + _delta < _oldValue`.

Recommendation

We advise rewriting the function accordingly to the desired logic.

For example, if this function is supposed to handle integer overflow then the statement could be rewritten as follow:

```
        if (_oldValue + _delta < _oldValue || _oldValue + _delta) = MAX_UINT256) {
            allowed[msg.sender][_spender] = MAX_UINT256.sub(1);
```

HMT-15 | Inconsistency Between Comment And Code

Category	Severity	Location	Status
Inconsistency	Discussion	Escrow.sol (a4a792c): 166~168; Escrow.sol (8d2a8d7): 159~161	Pending

Description

In the contract `Escrow.sol` the function `bulkPayOut()` perform a check on two input parameters as follow:

```
159         bool writeOnchain = bytes(_hash).length != 0 || bytes(_url).length != 0;
160         if (writeOnchain) {
161             // Be sure they are both zero if one of them is
162             finalResultsUrl = _url;
163             finalResultsHash = _hash;
164         }
```

However, the comment seems inconsistent with this check, indeed the conditional statement will not execute the block of code only if at most one of the two parameters is equal to zero. Then, if only one parameter is equal to zero, nothing in the logic of the function ensures that:

```
            // Be sure they are both zero if one of them is
```

Recommendation

We advise correcting the comment or rewriting the function so the logic and the comment describing it are consistent.

HMT-16 | Unclear Purpose Of `storeResults()`

Category	Severity	Location	Status
Gas Optimization	Discussion	Escrow.sol (a4a792c): 134~141; Escrow.sol (8d2a8d7): 131~138	Pending

Description

The function `storeResults()` check if the caller is trusted, if the contract is not expired, if the current status is `Pending` or `Partial` and then will emit an event with the two inputs parameter.

Moreover, no other function in the contract calls it.

Recommendation

We advise checking if the function is really necessary and removing it if not.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

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 you (“Customer” or 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, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

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 CertiK 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. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND

“AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.