CERTIK

Security Assessment

Hashland

Nov 3rd, 2021

Table of Contents

Summary

This report has been prepared for Hashland to discover issues and vulnerabilities in the source code of the Hashland 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	Hashland
Platform	ethereum
Language	Solidity
Codebase	https://github.com/HashlandGamefi/hashland-core/tree/main/contracts
Commit	34a853c9df37d41b33f96db0f46180867ef3bcf2

Audit Summary

Delivery Date	Nov 03, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

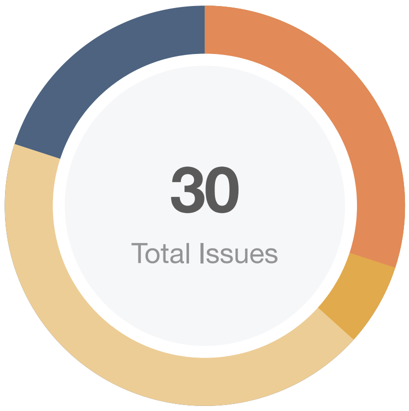
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● Major	9	0	0	8	0	1
● Medium	2	0	0	2	0	0
● Minor	13	0	0	4	0	9
● Informational	6	0	0	1	0	5
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
IHN	pool/interface/IHNMarket.sol	54e9d742679867a28b3ae3156826220b2d2064e647512cb29f2d79a59926f3ca
IHP	pool/interface/IHNPool.sol	afcfea06227287fd67009ff75be6a413fe29fe4dcff435d619bf182d0bc1de00
IIP	pool/interface/IInvitePool.sol	7e06e12a65e9ee394c74bff06d6b67c3f0d48ca2f9e4531fbb41d3286dd78c2e
HCL	pool/HCLPPool.sol	83a44804714ad579079370adf4f19620c8f9d4cb4d81ec8e83cff69689ca6979
HNB	pool/HNBox.sol	284fcae6fb1851cf2332e443548b33e8c1b9b8c4d2cd4df43b28994b1bfd227c
HNM	pool/HNMarket.sol	3d20bf551030c60030d57a8bdf5d8655f1929c04b4d4d545074ab21fcb6343a9
HNP	pool/HNPool.sol	5962a009ef0d5812894cdb49329415a8e6fc2e4abcd413dfb5e2fd657b604eaa
HNU	pool/HNUpgrade.sol	6f8f5afa174341de6b8020f3b91faf96b83e275591846bc1fb15d4a77820312d
IPH	pool/InvitePool.sol	2ca46618167733c361072033bb5660c2cea0cccccbbd7a3d09c3e71e770f71c3
IHC	token/interface/IHC.sol	dd9a1665f460fe250a42acabd0766bcfd21d0f356d0bbdcb6a07df687da1d161
IHH	token/interface/IHN.sol	9c6f6cb708e41cb358ef8bf147e954b8667bacb9279dd322ec91ea873a012048
HCH	token/HC.sol	48cddee4b38c32be22b09de6b148e020653583da22839db635af7c4f21eca4ad
HNH	token/HN.sol	4de841095c84db6beef54b08895822e62d95ffd430c5480eb3db0ff68c33a4d7

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**9** (30.00%)
🟨	**Medium**	**2** (6.67%)
🟨	**Minor**	**13** (43.33%)
🟦	**Informational**	**6** (20.00%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Financial Models	Data Flow, Control Flow	● Informational	ⓘ Acknowledged
HCH-01	Initial token distribution	**Centralization / Privilege**	● **Medium**	ⓘ Acknowledged
HCH-02	Lack of sanity check in function `subWeight()`	Volatile Code	● Informational	✓ Resolved
HCH-03	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
HCH-04	Missing emit events	Coding Style	● Informational	✓ Resolved
HCL-01	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Minor	✓ Resolved
HCL-02	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
HNB-01	Lack of sanity check in function `buyBoxes()`	Logical Issue	● Minor	✓ Resolved
HNB-02	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Minor	✓ Resolved
HNB-03	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
HNB-04	Admin can mint tokens for free	**Centralization / Privilege**	● **Major**	✓ Resolved

ID	Title	Category	Severity	Status
HNB-05	Function `adminBuyBoxes()` does not accumulate `totalBoxesLength`	Logical Issue	● Informational	Resolved
HNB-06	Weak pseudo random number generator	Logical Issue	● Minor	Acknowledged
HNH-01	Weak pseudo random number generator	Logical Issue	● Minor	Acknowledged
HNH-02	Centralization Risk	**Centralization / Privilege**	● **Major**	Acknowledged
HNM-01	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Minor	Resolved
HNM-02	Centralization Risk	**Centralization / Privilege**	● **Major**	Acknowledged
HNM-03	No upper limit for `feeRate`	Logical Issue	● Minor	Resolved
HNM-04	Logical issue of `sellers`	Logical Issue	● Informational	Resolved
HNP-01	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Minor	Resolved
HNP-02	Logical issue of function `airdropTokens()`	Logical Issue	● Minor	Acknowledged
HNP-03	Centralization Risk	**Centralization / Privilege**	● **Major**	Acknowledged
HNP-04	Logical issue of function `setMaxSlots()`	Logical Issue	● Informational	Resolved
HNP-05	Insufficient Reward Distribution	Logical Issue	● Medium	Acknowledged
HNU-01	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Minor	Resolved
HNU-02	Centralization Risk	**Centralization / Privilege**	● **Major**	Acknowledged
HNU-03	Function `upgrade()` does not check the ownership of material tokens	Logical Issue	● Minor	Resolved
HNU-04	Weak pseudo random number generator	Logical Issue	● Minor	Acknowledged
IPH-01	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	● Minor	Resolved

ID	Title	Category	Severity	Status
IPH-02	Centralization Risk	Centralization / Privilege	● Major	ⓘ Acknowledged

GLOBAL-01 | Financial Models

Category	Severity	Location	Status
Data Flow, Control Flow	● Informational	Global	ⓘ Acknowledged

Description

Financial models of blockchain protocols need to be resilient to attacks. It needs to pass simulations and verifications to guarantee the security of the overall protocol. The financial model of this protocol is not in the scope of this audit.

Alleviation

The team acknowledged this issue and they stated the following:

"They have hired a professional financial model analysis team to monitor and analyze the operation data of our projects throughout the process."

HCH-01 | Initial token distribution

Category	Severity	Location	Status
Centralization / Privilege	● **Medium**	token/HC.sol: 50	ⓘ Acknowledged

Description

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

Recommendation

We recommend the team to make the token allocation plan transparent.

Alleviation

The team acknowledged this issue and they stated the following:

"According to their white paper, the total amount of HC tokens is 21 million. When deploying the contract, 2.1 million will be pre-mined to the multi-signature wallet of the board of directors."

HCH-02 | Lack of sanity check in function `subWeight()`

Category	Severity	Location	Status
Volatile Code	● Informational	token/HC.sol: 87	✓ Resolved

Description

The function `subWeight()` does not check if `block.number >= startBlock` to ensure the project has started.

Recommendation

We recommend the team adding the checking like what is checked in the function `addWeight()`.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HCH-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	token/HC.sol	ⓘ Acknowledged

Description

In the contract `HC`, the role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

The role `MANAGER_ROLE` has the authority over the following functions:

- `addWeight()` and `subWeight()` to set poolWeight.

Any compromise to the above-mentioned roles may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `DEFAULT_ADMIN_ROLE` and `MANAGER_ROLE` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE and MANAGER_ROLE."

HCH-04 | Missing emit events

Category	Severity	Location	Status
Coding Style	● Informational	token/HC.sol	✓ Resolved

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to users.

- `HC.addWeight()`
- `HC.subWeight()`
- `HC.harvestToken()`
- `HN.spawnHn()`
- `HN.setLevel()`
- `HN.setHashrates()`
- `HN.setData()`
- `HCLPPool.setOpenStatus()`
- `HNBox.setTokensInfo()`
- `HNBox.setReceivingAddress()`
- `HNBox.addBoxesMaxSupply()`
- `HNBox.setDatas()`
- `HNMarket.setOpenStatus()`
- `HNMarket.setFeeRatio()`
- `HNMarket.setReceivingAddress()`
- `HNPool.setTokensInfo()`
- `HNPool.setOpenStatus()`

- `HNPool.setMaxSlots()`
- `HNPool.setSlotBasePrice()`
- `HNPool.setReceivingAddress()`
- `HNPool.setInvitePoolAddress()`
- `HNPool.setHNMarketAddress()`
- `HNPool.airdropTokens()`
- `HNUpgrade.setMaxLevel()`
- `HNUpgrade.setUpgradeBasePrice()`
- `HNUpgrade.setReceivingAddress()`
- `HNUpgrade.setDatas()`

Recommendation

We recommend the client add events for sensitive actions, and emit them in the function.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HCL-01 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Minor	pool/HCLPPool.sol: 82, 113, 137	✓ Resolved

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

The aforementioned lines perform external call to `transferFrom` of ERC20 contracts and the return value is not checked in either case.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HCL-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	pool/HCLPPool.sol	ⓘ Acknowledged

Description

In the contract `HCLPPool`, the role `MANAGER_ROLE` has the authority over the following function.

- `setOpenStatus()`, to change `openStatus`.

The role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

Any compromise to the above-mentioned roles may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `MANAGER_ROLE` or `DEFAULT_ADMIN_ROLE` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE and MANAGER_ROLE."

HNB-01 | Lack of sanity check in function `buyBoxes()`

Category	Severity	Location	Status
Logical Issue	● Minor	pool/HNBox.sol: 120	✓ Resolved

Description

The function `buyBoxes` does not check if `price[tokenId] > 0` to ensure the project support the certain token.

Checking `tokenId < tokenAddrs.length` is helpful as well.

Recommendation

We recommend the team add the necessary check for the parameter `tokenId`.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNB-02 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Minor	pool/HNBox.sol: 135	✓ Resolved

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

The aforementioned lines perform external call to `transferFrom` of ERC20 contracts and the return value is not checked in either case.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNB-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	pool/HNBox.sol	ⓘ Acknowledged

Description

In the contract `HNBox`, the role `MANAGER_ROLE` has the authority over the following function:

- `setTokensInfo()`, to set `boxTokenPrices` and `tokenAddrs`.
- `setReceivingAddress()`, to set `receivingAddress`.
- `addBoxesMaxSupply()`, to add the `boxesMaxSupply`.
- `setDatas()`, to set `btcBase` and `btcRange`.
- `adminBuyBoxes()`, to buy free HN tokens.

The role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

Any compromise to the `MANAGER_ROLE` and `DEFAULT_ADMIN_ROLE` accounts may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `MANAGER_ROLE` and `DEFAULT_ADMIN_ROLE` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE and MANAGER_ROLE."

HNB-04 | Admin can mint tokens for free

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	pool/HNBox.sol: 107	✓ Resolved

Description

The function `adminBuyBoxes()` allows accounts with `MANAGER_ROLE` to mint tokens for free.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNB-05 | Function `adminBuyBoxes()` does not accumulate `totalBoxesLength`

Category	Severity	Location	Status
Logical Issue	● Informational	pool/HNBox.sol: 107	✓ Resolved

Description

The function `adminBuyBoxes()` does not accumulate `totalBoxesLength`.We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNB-06 | Weak pseudo random number generator

Category	Severity	Location	Status
Logical Issue	● Minor	pool/HNBox.sol: 196~226	ⓘ Acknowledged

Description

The `random` number generation is based on the result of `keccak256` encryption of data that is easy to control or get. The result can be controlled by the caller of the transaction.

Recommendation

We recommend using a verifiable source of randomness, such as Chainlink VRF, for the random number generation.

Alleviation

The team acknowledged this issue and they stated the following:

"This is our basic blind box contract. The unit price of each card is very low. If Chainlink VRF is used, the cost of our project will increase significantly, so this random number is designed to change according to the number of cards sold. Because our blind box cards will be sold very fast, they will be sold out within a few minutes, so the possibility of being attacked is very low. We will use Chainlink VRF in future advanced blind box contracts. Because the card unit price of advanced blind box is relatively high, using Chainlink VRF will not increase the cost significantly."

HNH-01 | Weak pseudo random number generator

Category	Severity	Location	Status
Logical Issue	● Minor	token/HN.sol: 168	ⓘ Acknowledged

Description

The `random` number generation is based on the result of `keccak256` encryption of data that is easy to control or get. The result can be controlled by the caller of the transaction.

Recommendation

We recommend using a verifiable source of randomness, such as Chainlink VRF, for the random number generation.

Alleviation

The team acknowledged the issue and stated the following.

"This function is designed to generate random numbers based on hnId and slot. For example, the class attribute(range 1-4) of the NFT with ID 3 is always 2."

HNH-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	token/HN.sol	ⓘ Acknowledged

Description

In the contract `HN`, the role `SPAWNER_ROLE` has the authority over the following function:

- `spawnHN()`, to generate a new `HN` token.

The role `SETTER_ROLE` has the authority over the following functions:

- `setLevel()`, to set level for the token.
- `setHashrates()`, to set hashrates for the token.
- `setData()`, to set data for the token.
- `setDatas()`, to set datas for the token.

The role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

Any compromise to the above-mentioned roles may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the SPAWNER_ROLE`, `SETTER_ROLE`, `DEFAULT_ADMIN_ROLE` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE and MANAGER_ROLE. SPAWNER_ROLE will be granted to the HNBox contract. SETTER_ROLE will be granted to the HNUpgrade contract."

HNM-01 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Minor	pool/HNMarket.sol: 213~214	✓ Resolved

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

The aforementioned lines perform external call to `transferFrom` of ERC20 contracts and the return value is not checked in either case.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNM-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	pool/HNMarket.sol	ⓘ Acknowledged

Description

In the contract `HNMarket`, the role `MANAGER_ROLE` has the authority over the following functions:

- `setOpenStatus()`, to set `openStatus`.
- `setFeeRatio()`, to set `feeRatio`.
- `setReceivingAddress()`, to set `receivingAddress`.

The role `HNPOOL_ROLE` has the authority over the following functions:

- `hnPoolCancel()`, to cancel orders.

The role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

Any compromise to the `MANAGER_ROLE`, `DEFAULT_ADMIN_ROLE` and `HNPOOL_ROLE` accounts may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `MANAGER_ROLE`, `DEFAULT_ADMIN_ROLE` and `HNPOOL_ROLE` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE and MANAGER_ROLE. HNPOOL_ROLE will be granted to the HNPool contract."

HNM-03 | No upper limit for `feeRate`

Category	Severity	Location	Status
Logical Issue	● Minor	pool/HNMarket.sol: 105	✓ Resolved

Description

The owner can call the `setFeeRate()` function to set the `feeRate` and there is no upper limit on what the rate can be. In the extreme case, the rate can be as high as 100%, which would imply that reward pools cannot get any token back after calling the `withdraw()` function.

Recommendation

We recommend setting a reasonable upper limit for the `feeRate` variable, such as 10%, 20% or 50% are more reasonable than 100%.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNM-04 | Logical issue of `sellers`

Category	Severity	Location	Status
Logical Issue	● Informational	pool/HNMarket.sol: 45	✓ Resolved

Description

The user will be added to the `sellers` when the function `sell()` is called. But the user will not be removed from the `sellers` when the function `cancel()` is called.

```
EnumerableSet.AddressSet private sellers;
function sell(){
...
  sellers.add(msg.sender);
...
}
```

Recommendation

We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The team heeded our advice and resolved this issue in commit `0c6a5f1b7636feba1efec0a1b227fd61216c89bc`.

HNP-01 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Minor	pool/HNPool.sol: 180, 366, 384	✓ Resolved

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

The aforementioned lines perform external call to `transferFrom` of ERC20 contracts and the return value is not checked in either case.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNP-02 | Logical issue of function `airdropTokens()`

Category	Severity	Location	Status
Logical Issue	● Minor	pool/HNPool.sol: 181	ⓘ Acknowledged

Description

The calculation of `tokensPerBlock` is wrong when the function is called more than once. Because total airdrop tokens contain the parameter amount and some leftover.

Alleviation

The team acknowledged the issue and stated the following.

They will ensure that tokens will only be airdropped once a day based on the `lastAirdropTimes`.

HNP-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	pool/HNPool.sol	ⓘ Acknowledged

Description

In the contract `HNPool`, the role `MANAGER_ROLE` has the authority over the following function:

- `setTokensInfo()`, to set `tokenAddrs` and `tokensPerBlock`.
- `setOpenStatus()`, to set `openStatus`.
- `setMaxSlots()`, to set `maxSlots`.
- `setSlotBasePrice()`, to set `slotBasePrice`.
- `setReceivingAddress()`, to set `receivingAddress`.
- `setInvitePoolAddress()`, to set `invitePool`.
- `setHNMarketAddress()`, to set `hnMarket`.
- `airdropTokens()`, to airdrop tokens.

The role `hnMarket` has the authority over the following function:

- `hnMarketWithdraw()`, to withdraw tokens.

The role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

Any compromise to the `DEFAULT_ADMIN_ROLE`, `MANAGER_ROLE` and `hnMarket` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `DEFAULT_ADMIN_ROLE`, `MANAGER_ROLE` and `hnMarket` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE, MANAGER_ROLE and AIRDROPPER_ROLE. hnMarket will be granted to the HNMarket contract."

HNP-04 | Logical issue of function `setMaxSlots()`

Category	Severity	Location	Status
Logical Issue	● Informational	pool/HNPool.sol: 118	✓ Resolved

Description

A decrease of the `maxSlots` may lead to underflow errors in function `getUserLeftSlots()`.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNP-05 | Insufficient Reward Distribution

Category	Severity	Location	Status
Logical Issue	● Medium	pool/HNPool.sol: 511	ⓘ Acknowledged

Description

The function `updatePool()` doesn't check if the amount of airdrop tokens is sufficient or the releaseBlocks limit set in the function `airdropTokens()`.

The balance of the contract might be insufficient to cover the cost of rewards distribution.

Alleviation

The team acknowledged the issue and stated the following.

"Because the time of the daily airdrop of tokens cannot be guaranteed to always be the same, it is necessary to give priority to ensuring that the user's daily income is constant. We will ensure that tokens are airdropped once a day."

HNU-01 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Minor	pool/HNUpgrade.sol: 118, 139	✓ Resolved

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

The aforementioned lines perform external call to `transferFrom` of ERC20 contracts and the return value is not checked in either case.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNU-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	pool/HNUpgrade.sol	ⓘ Acknowledged

Description

In the contract `HNUpgrade`, the role `MANAGER_ROLE` has the authority over the following function:

- `setMaxLevel()`, to set `maxLevel`.
- `setUpgradeBasePrice()`, to set `upgradeBasePrice`.
- `setReceivingAddress()`, to set `receivingAddress`.
- `setDatas()`, to set `hcBase`, `hcRange`, `hashratesBase` and `hashratesRange`.

The role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

Any compromise to the `MANAGER_ROLE` and `DEFAULT_ADMIN_ROLE` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `MANAGER_ROLE` and `DEFAULT_ADMIN_ROLE` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE and MANAGER_ROLE."

HNU-03 | Function `upgrade()` does not check the ownership of material tokens

Category	Severity	Location	Status
Logical Issue	● Minor	pool/HNUpgrade.sol: 133	✓ Resolved

Description

The function `upgrade()` does not check the ownership of the material tokens. We would like to confirm with the client if the current implementation aligns with the original project design.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

HNU-04 | Weak pseudo random number generator

Category	Severity	Location	Status
Logical Issue	● Minor	pool/HNUpgrade.sol: 164~179	ⓘ Acknowledged

Description

The `random` number generation is based on the result of `keccak256` encryption of data that is easy to control or get. The result can be controlled by the caller of the transaction.

Recommendation

We recommend using a verifiable source of randomness, such as Chainlink VRF, for the random number generation.

Alleviation

The team acknowledged this issue and they stated the following:

"This is our card synthesis upgrade contract. The HC Token consumed per synthesis is not too much. If Chainlink VRF is used, it will significantly increase the cost of our project. Because the rule of synthesis upgrade is that the more cards of the same class, the more hashrate will be increased, so the influence of random numbers is very small. Even if it is attacked, the impact on the project is not too great. We will open more advanced synthetic upgrade contracts in the future. A single upgrade will consume a lot of resources. At this time, Chainlink VRF will be used, which will not significantly increase project costs."

IPH-01 | Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	● Minor	pool/InvitePool.sol: 183	✓ Resolved

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of a proper ERC-20 implementation.

The aforementioned lines perform external call to `transferFrom` of ERC20 contracts and the return value is not checked in either case.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from 'transferFrom()' is checked.

Alleviation

The team heeded our advice and fixed the issue in commit 798ee6cefb77aad09a1d6a01dfa06437b40ead89.

IPH-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	pool/InvitePool.sol	ⓘ Acknowledged

Description

In the contract `InvitePool`, the role `MANAGER_ROLE` has the authority over the following function:

- `setOpenStatus()`, to set `openStatus`.

The role `HNPOOL_ROLE` has the authority over the following function:

- `depositInviter()` and `withdrawInviter()`, to change `inviterStake` for the user.

The role `DEFAULT_ADMIN_ROLE` has the authority over the following functions:

- `AccessControlEnumerable.grantRole()` and `AccessControlEnumerable.revokeRole()` to grant role or revoke role.

Any compromise to the `MANAGER_ROLE`, `HNPOOL_ROLE` and `DEFAULT_ADMIN_ROLE` account may allow the hacker to take advantage of this.

Recommendation

We advise the client to carefully manage the `MANAGER_ROLE`, `HNPOOL_ROLE` and `DEFAULT_ADMIN_ROLE` accounts' private keys to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

The team acknowledged this issue and they stated the following:

"They will use a multi-signature wallet to manage DEFAULT_ADMIN_ROLE and MANAGER_ROLE. HNPOOL_ROLE will be granted to the HNPool contract."

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.

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.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

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.

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.