// HALBORN

Access Protocol - Cairo contracts

Cairo Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **July 25th, 2022 - August 22nd, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/08/2022	Gokberk Gulgun
0.2	Document Edits	08/15/2022	Gokberk Gulgun
1.0	Remediation Plan	08/29/2022	Gokberk Gulgun
1.1	Remediation Plan Review	08/29/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Access Protocol engaged Halborn to conduct a security audit on their smart contracts beginning on July 25th, 2022 and ending on August 22nd, 2022 . The security assessment was scoped to the smart contracts provided to the Halborn team.

Access Protocol is a protocol that allows content creators to earn rewards whenever a user wants to access their content. Each content creator would create a stake pool, and based on how much ACCESS tokens are staked in their pool, they would get a percentage of the reward issued from the protocol daily inflation. Users, who wants to access the content, must first deposit a stake in the pool. Users are also rewarded a portion of the daily inflation.

1.2 AUDIT SUMMARY

The team at Halborn was provided six weeks for the engagement and assigned two full-time security engineers to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended.
- Identify potential security issues with the smart contracts.

In summary, Halborn identified several security risks which were mostly addressed by the Access Protocol team.

Halborn identified that malicious content creators could trick the users to stake tokens in their pool for a certain reward percentage (called stakers_part), however, as rewards are calculated daily, the pool owners would be able to change this parameter to be a more favorable value (up to

1% for stakers and 99% for pool owners). This change could be performed just before the day end (calculated using the block.timestamp), and then, after the new day start, the pool owner could get 99% of the reward for themselves.

- Malicious usage.
- Access control (the platform is centralized, and it might require adding additional user roles).
- Lack of documentation and inconsistencies in code/comments.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy regarding the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation, automated testing techniques help enhance coverage of the protocol code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions.
- Manual assessment of use and safety for the critical Cairo variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing using custom scripts.
- Testnet deployment (starknet-devnet).
- Dynamic testing with Protostar and Nile and custom Python scripts.

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.
3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

The contracts that were audited can be found in the following repository:

Access Protocol Cairo contracts.

The commit ID provided for the assessment was:

- 927e9a072c8e11485ebec19ed96b0139c4ba0fb1

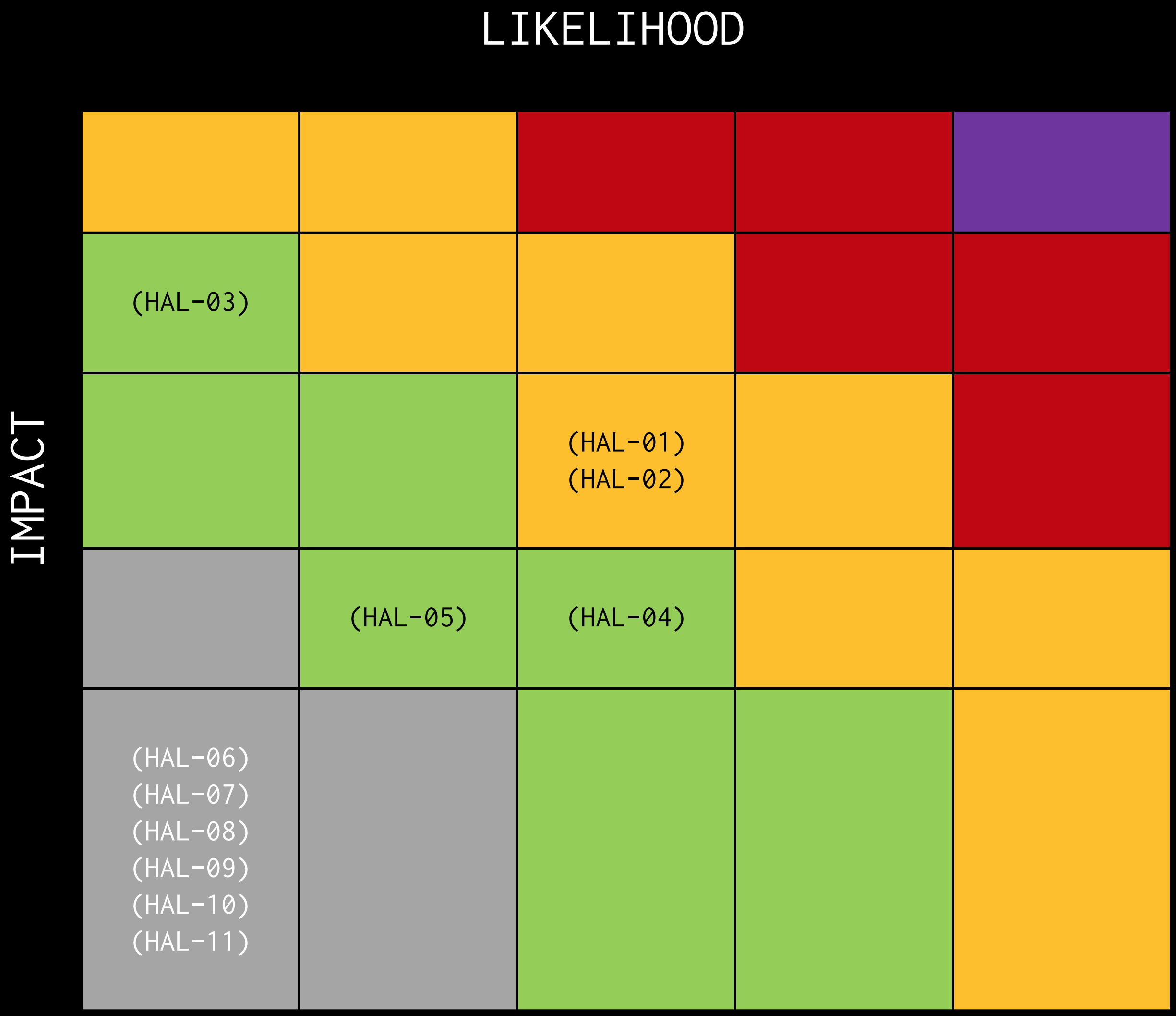
Commit IDs for new changes were :

- PR 75
- PR 76
- PR 77

Please note that the OpenZeppelin's contracts, which are in use by the project, were not within scope for this assessment. Furthermore, these have not been audited and should be used with care.

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	2	3	6



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL-01 - STAKE POOL OWNER CAN CHANGE THE REWARDS EARNED	Medium	SOLVED - 08/29/2022
HAL-02 - OVER PRIVILEGED AUTHORITY ROLE	Medium	RISK ACCEPTED
HAL-03 - MISSING MECHANISM FOR FORCED EXIT ON FROZEN POOL	Low	NOT APPLICABLE - 08/29/2022
HAL-04 - INSUFFICIENT PARAMETER VALIDATION	Low	RISK ACCEPTED
HAL-05 - MISSING TWO-STEP TRANSFER OWNERSHIP PATTERN	Low	RISK ACCEPTED
HAL-06 - OUTDATED OPENZEPPELIN'S CONTRACTS VERSION	Informational	SOLVED - 08/29/2022
HAL-07 - CONTRACTS WITH THE SAME NAME WILL HAVE OVERWRITTEN ABIS	Informational	ACKNOWLEDGED
HAL-08 - UNCLEAR VARIABLE NAMING	Informational	SOLVED - 08/29/2022
HAL-09 - CODE COMMENTS INCONSISTENCIES	Informational	SOLVED - 08/29/2022
HAL-10 - WRONG ARRAY DECLARATION FOR FUNCTION PARAMETERS	Informational	SOLVED - 08/29/2022
HAL-11 - UNUSED VARIABLE	Informational	SOLVED - 08/29/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) STAKE POOL OWNER CAN CHANGE THE REWARDS EARNED - MEDIUM

Description:

The owner of a pool stake can change the stakers_part parameter, which is the percentage of the pool reward goes to staking users compared to the pool owner. The only validation on staker_part is done with the following function:

Listing 1

```
func assert_valid_stakers_part{range_check_ptr}(stakers_part: felt
↳ ) -> ():
    with_attr error_message("stakers_part must be in [0, 100]
↳ range"):
        assert_nn_le(stakers_part, 100)
    end
    return ()
end
```

The above function only validates that the staker's share is between 0 and 100, so it can be set to as little as 1%. Due to how the funds are locked in the pool, requiring users to not be able to unstake until a day has passed, a malicious pool owner could take the following steps:

1. Create a legitimate stake pool.
2. Get pool approval from AccessProtocol.
3. Wait for users to deposit their stake.
4. At the end of the day, when the rewards are available, change the stakers part to be just 1%.
5. Withdraw rewards from the pool.

Proof of Concept:

1. call create_stake_pool.
2. The administrator would approve the pool.

3. The creator of the malicious pool waits for users to stake tokens.
4. The malicious pool owner changes the staker part to be 1%.
5. One day, passes when rewards can be claimed.
6. The owner of the malicious pool withdraws his reward percentage.

Users can now identify the issue and report it to Access Protocol, at which point they can freeze the pool so that the owner cannot claim further interest. However, this would result in the staked funds being locked in the pool (as described in the next issue). If Access Protocol were to attempt to compensate stakeholders, they could mint an arbitrary amount of tokens, however the total supply of the token would increase, affecting all holders.

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Halborn recommends reviewing the staking functionality as well as the pool administration process. The first step of the actions would be to limit the number of times the stakers_part can be changed within a certain period of time, for instance once a day would be sufficient and would not require a large number of code changes. Furthermore, users should be aware of this potential risk in the interface, to prevent them from locking their tokens in a "malicious" pool. An additional safeguard mechanism for users would be to allow their funds to unstake in case the owner of the pool changes the stakers_part beyond a certain threshold. This can be achieved by storing the stakers_part variable at the stake time and implementing an unstake function that checks if the contract's current stakers_part has changed by more than a pre-set value.

Another possible solution to protect users could involve creating two types of pool, one that allows the owner of the pool to change the stakers_parts, and one that is fixed. The pool that allows the reward percentage to be modified should implement the mitigation described above, while the fixed pool should be set with the reward set to a value that the Access Protocol team deems appropriate.

Remediation Plan:

SOLVED: The Access Protocol team implemented changes in the following PR 75 and PR 76.

3.2 (HAL-02) OVER PRIVILEGED AUTHORITY ROLE - MEDIUM

Description:

The protocol authority role has many privileges on the system. For instance, they can mint new tokens, freeze and unfreeze accounts, approve or disable staking pools, change token emission parameters.

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Halborn recommends that the protocol be reviewed to ensure that a role-based access control system is implemented, to reduce the damage an attacker could cause with authority account access. At a minimum, the Access Protocol team should ensure that the role with the highest privilege uses a multi-signature account, or if this is not possible (for instance, the backend system needs to submit transactions on their behalf), it should be considered separating user roles, be more granular.

Remediation Plan:

RISK ACCEPTED: The Access Protocol team will implement multi-sig solution when there is a stable solution.

3.3 (HAL-03) MISSING MECHANISM FOR FORCED EXIT ON FROZEN POOL - LOW

Description:

The protocol admin account can freeze pools in case they detect malicious behavior; however, there is no process to allow users to opt out of their stake.

Unstake-related functions, such as queue_unstake and execute_unstake check if the pool is in active state; therefore, legitimate users cannot leave a frozen pool.

The Access Protocol team confirmed that this pool freeze functionality will only be used in the event of a pool exploit, thus reducing the likelihood of a user having their funds locked into a pool without an exit mechanism.

Code Location:

stake/interface.cairo

- execute_unstake
- queue_unstake

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

It is recommended to add a separate function that allows users to leave a frozen pool. This should verify that the caller is one of the stakers, and not the pool owner, in case the pool owner has staked tokens in their own pool, they should not be allowed out.

Remediation Plan:

NOT APPLICABLE: The Access Protocol team states that the only reason to freeze a pool is when it is created for malicious reasons, and then all parties involved should freeze their funds as well. In case someone wants to punish the owner of the pool, then the owner can freeze himself/herself, allowing the stakers to withdraw their funds.

3.4 (HAL-04) INSUFFICIENT PARAMETER VALIDATION - LOW

Description:

Certain administrative functionality does not impose limits on the values passed as arguments. For instance, the authority role could change the token inflation to zero at any time, possibly affecting the token rewards. If this parameter is changed to zero or a very low value, users with staked asset will not be able to earn rewards and will eventually end up with their funds locked for the minimum unlock time.

Code Location:

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Halborn recommends that these functions be reviewed to ensure that values entered by the authority role are within certain limits.

Remediation Plan:

RISK ACCEPTED: The Access Protocol team accepted the risk of this finding.

3.5 (HAL-05) MISSING TWO-STEP TRANSFER OWNERSHIP PATTERN - LOW

Description:

The code does not implement a two-step ownership transfer pattern. This practice is recommended when admin users have heavy responsibilities, such as the ability to mint tokens, freeze or unfreeze user accounts and set system configurations.

It may happen that when transferring ownership of a contract, an error is made in the address. If the request were submitted, the contract would be lost forever. With this pattern, contract owners can submit a transfer request; however, this is not final until accepted by the new owner. If they realize they have made a mistake, they can stop it at any time before accepting it by calling `cancelRequest`.

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Halborn recommends that the pattern be implemented to ensure that property transfer errors are not permanent. An implementation of this can be found in the OpenZeppelin's cairo-contracts pull requests at GitHub. Note that this has not yet been accepted, as the repository maintainers are waiting for the Solidity release to be completed first, so it can follow the same structure.

Furthermore, this still allows contract owners to renounce ownership if they wish, via the `renounceOwnership` function.

Remediation Plan:

RISK ACCEPTED: The Access Protocol team will implement it when the two-step-transfer ownership exists in the stable version of OZ contracts.

3.6 (HAL-06) OUTDATED OPENZEPPELIN'S CONTRACTS VERSION - INFORMATIONAL

Description:

A software component is part of a system or application that extends the functionality of the application, such as a module, software package. Because the Cairo language is still in its early days of development, any third-party library should be thoroughly tested to ensure that no unintended bugs are introduced to the platform.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Halborn recommends always using the lastest and most stable version of third-party dependencies. Furthermore, because the Cairo language is still in its early days of development, any third-party library should be thoroughly tested to ensure that no unintended bugs are introduced to the platform. The current version of the contracts is v0.3.0 available here.

Remediation Plan:

SOLVED: The Access Protocol team implemented changes in the following PR 78

3.7 (HAL-07) CONTRACTS WITH THE SAME NAME WILL HAVE OVERWRITTEN ABIS - INFORMATIONAL

Description:

Multiple contracts share the same filename, and when compiled, the ABIs will be saved based on the filename. When two contracts have the same name, the ABI of the last compiled contract will be the one saved to disk.
This could cause confusion when interacting with the contracts, developing a backend/frontend for them.

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Although the project is well organized, it is recommended to implement more sensible file names, for instance:

administration_events.cairo
stake_events.cairo

Remediation Plan:

ACKNOWLEDGED: The Access Protocol Team acknowledged this issue.

3.8 (HAL-08) UNCLEAR VARIABLE NAMING - INFORMATIONAL

Description:

While manually reviewing the code, Halborn identified that a parameter variable for assert_valid_status had an incorrect name. This could cause errors later when using that function.

Code Location:

pool/types.cairo Line #164

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Halborn recommends renaming the parameter to status, as it is a more representative name.

Remediation Plan:

SOLVED: The Access Protocol team implemented changes in the following PR 77

3.9 (HAL-09) CODE COMMENTS INCONSISTENCIES - INFORMATIONAL

Description:

The code lacks proper documentation and certain comments do not reflect the actual functionality of the contracts. This could lead to increased maintenance of the code, as well as bugs in case a different developer modifies the code.

Code Location:

bond/interface.cairo

Recommendation:

Halborn recommends that more complete documentation be produced for the project, as well as that comments in the code be modified to reflect the actual functionality of the protocol.

Remediation Plan:

SOLVED: The Access Protocol team solved the issue in the main branch.

3.10 (HAL-10) WRONG ARRAY DECLARATION FOR FUNCTION PARAMETERS - INFORMATIONAL

Description:

Within the administration/functions.cairo contract, the function add_bond_sellers takes an array of felt as arguments. As stated in the Cairo documentation, functions that accept arrays as parameters should accept them in the following way:

In this example, we are considering the array variable to be named as array_arg and to be of type felt*:

- array_arg_len: felt
- array_arg: felt*

As shown above, each array parameter must have a corresponding length parameter, which must precede the array. This could cause issues with the compiler.

Code Location:

contract/src/adminstration/functions.cairo Line #103

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Halborn recommends that the parameters be reversed so that the sellers_len parameter precedes the sellers array.

Remediation Plan:

SOLVED: The Access Protocol team solved the issue with the removal of the bond acceptance mechanism.

3.11 (HAL-11) UNUSED VARIABLE - INFORMATIONAL

Description:

In the following code section, the variable is assigned, but never used.

Code Location:

contract/src/pool/interface.cairo Line #96

Listing 2

```
let time = pool.data.pool_creation_time
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider removing the unused variable.

Remediation Plan:

SOLVED: The Access Protocol team implemented changes in the following PR 77.

3.12 Appendix I

Bond Sellers Considerations:

The following information is provided as information on the functionality of the bond seller. After a discussion with the Access Protocol team, Halborn states that the code can be simplified to better fit its needs, as well as the surrounding documentation can be improved.

While contract deployers can set up multiple bond sellers, it was confirmed that only one seller would be defined during initialization, and this would be the authority account.

Consequently, the initializer function should be amended to accept only one bond seller, or set directly to be the authority's address (provided in the first parameter).

It is important that untrusted entities be not set up as bond sellers, as they could cause great damage to the platform.
Furthermore, being just a bond seller, the approval mechanism could be added to the bond creation function, to remove the need for a second function call.

On the other hand, if the Access Protocol team decides to have multiple bond sellers or extend this functionality to meet other needs, they should ensure that one bond seller cannot self approve their bond, as well as set themselves as the buyer. This would cause them to be able to mint new tokens at will. This could be achieved by checking the caller address to not be the bond creator, as well as implementing a larger BOND_SIGNER_THRESHOLD.

REMEDIATION

The Access Protocol team implemented some changes after a discussion about the bond sellers functionality, and decided to assign the role to the authority account. They will be using this to “airdrop” tokens to certain users by staking it for them so that they can earn a percentage of the reward from the pools without the ability to unstake them.

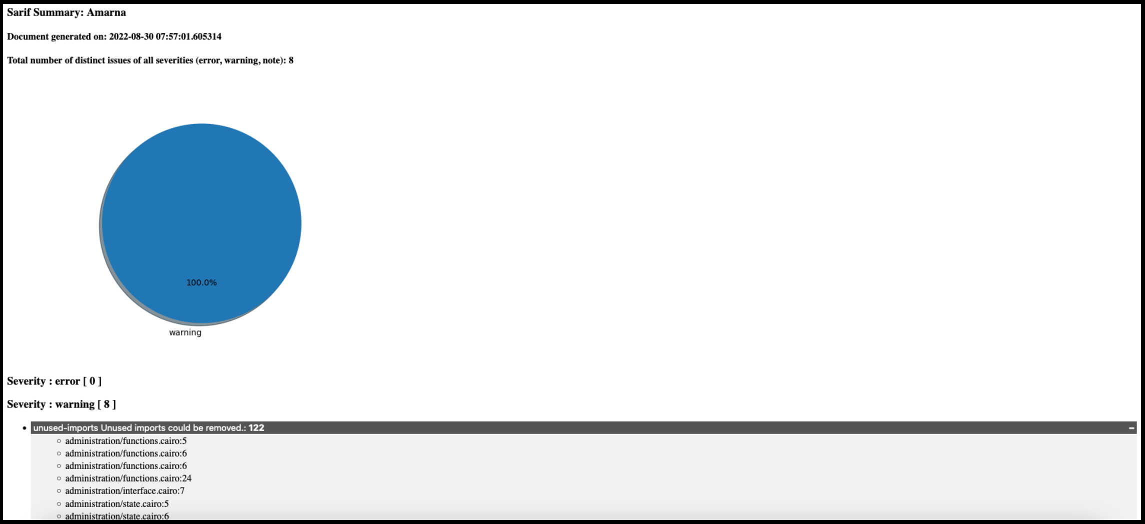
To support this change, they removed the parameters from the initializer and set the authority to be a bond seller.

STATIC ANALYSIS

Description:

Halborn used automated testing techniques to enhance coverage of certain areas of the scoped component. Among the tools used were amarna. After Halborn verified all the contracts and scoped structures in the repository and was able to compile them correctly, these tools were leveraged on scoped structures. With these tools, Halborn can statically verify security related issues across the entire codebase.

Amarna results:



Sarif Summary: Amarna

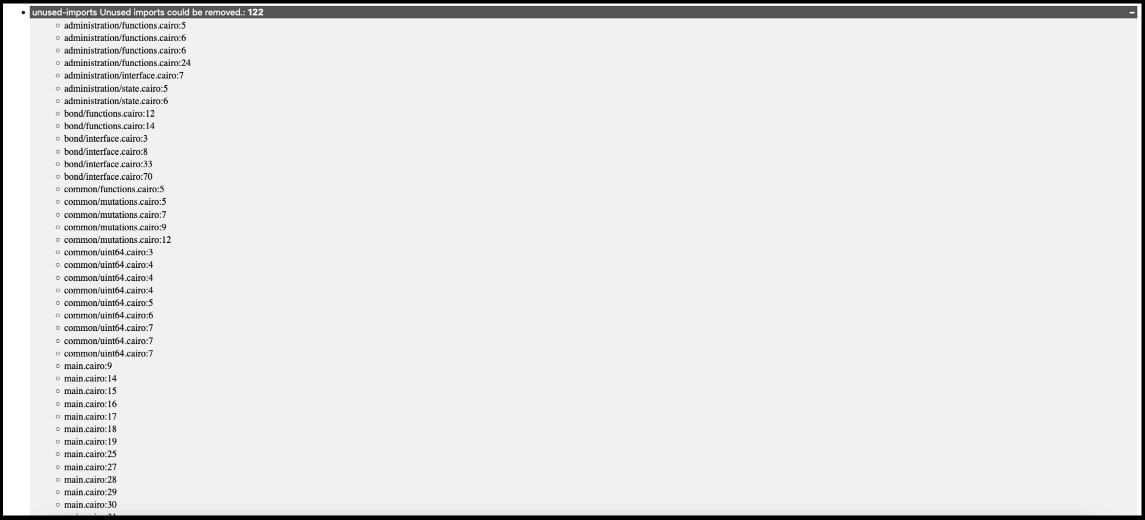
Document generated on: 2022-08-30 07:57:01.605314

Total number of distinct issues of all severities (error, warning, note): 8

Severity : error [0]

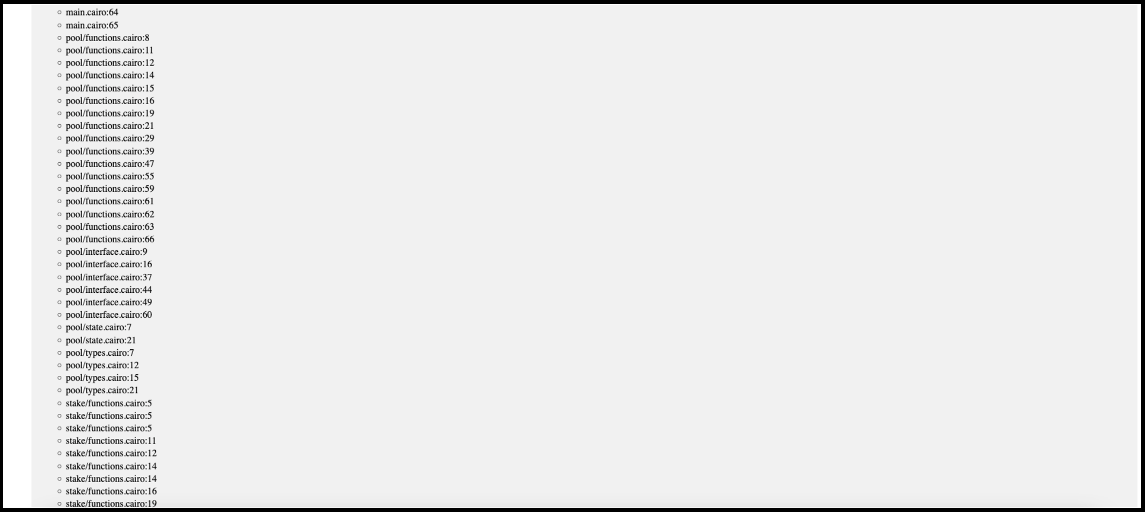
Severity : warning [8]

- unused-imports Unused imports could be removed.: **122**
  - administration/functions.cairo:5
  - administration/functions.cairo:6
  - administration/functions.cairo:6
  - administration/functions.cairo:24
  - administration/interface.cairo:7
  - administration/state.cairo:5
  - administration/state.cairo:6

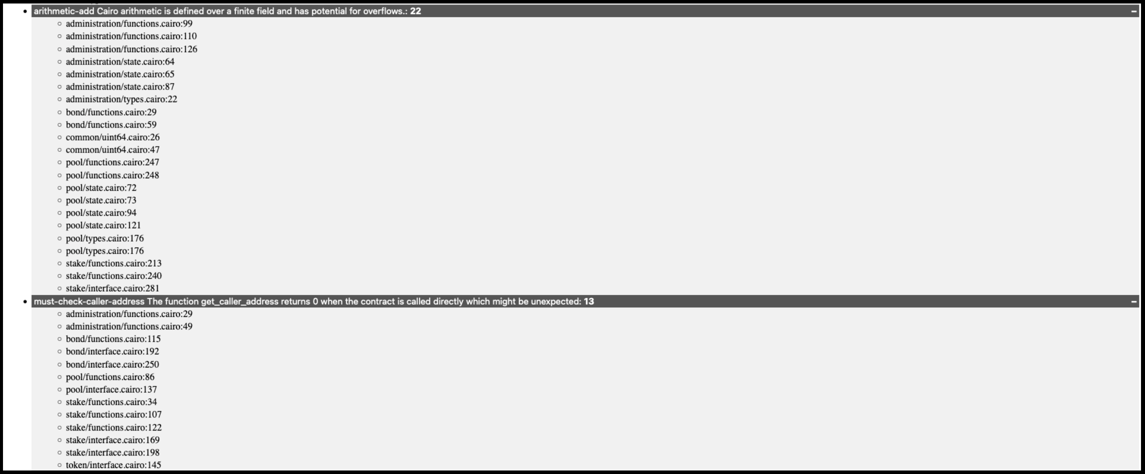


- unused-imports Unused imports could be removed.: **122**
  - administration/functions.cairo:5
  - administration/functions.cairo:6
  - administration/functions.cairo:6
  - administration/functions.cairo:24
  - administration/interface.cairo:7
  - administration/state.cairo:5
  - administration/state.cairo:6
  - bond/functions.cairo:12
  - bond/functions.cairo:14
  - bond/interface.cairo:3
  - bond/interface.cairo:8
  - bond/interface.cairo:33
  - bond/interface.cairo:70
  - common/functions.cairo:5
  - common/mutations.cairo:5
  - common/mutations.cairo:7
  - common/mutations.cairo:9
  - common/mutations.cairo:12
  - common/uint64.cairo:3
  - common/uint64.cairo:4
  - common/uint64.cairo:4
  - common/uint64.cairo:4
  - common/uint64.cairo:5
  - common/uint64.cairo:6
  - common/uint64.cairo:7
  - common/uint64.cairo:7
  - common/uint64.cairo:7
  - main.cairo:9
  - main.cairo:14
  - main.cairo:15
  - main.cairo:16
  - main.cairo:17
  - main.cairo:18
  - main.cairo:19
  - main.cairo:25
  - main.cairo:27
  - main.cairo:28
  - main.cairo:29
  - main.cairo:30

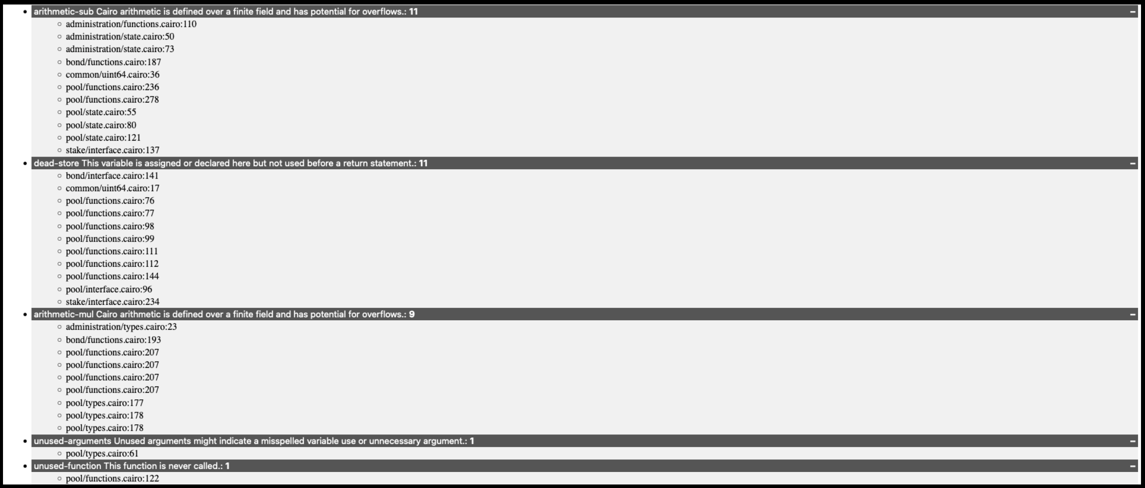
STATIC ANALYSIS



```
    ○ main.cairo:64
    ○ main.cairo:65
    ○ pool/functions.cairo:8
    ○ pool/functions.cairo:11
    ○ pool/functions.cairo:12
    ○ pool/functions.cairo:14
    ○ pool/functions.cairo:15
    ○ pool/functions.cairo:16
    ○ pool/functions.cairo:19
    ○ pool/functions.cairo:21
    ○ pool/functions.cairo:29
    ○ pool/functions.cairo:39
    ○ pool/functions.cairo:47
    ○ pool/functions.cairo:55
    ○ pool/functions.cairo:59
    ○ pool/functions.cairo:61
    ○ pool/functions.cairo:62
    ○ pool/functions.cairo:63
    ○ pool/functions.cairo:66
    ○ pool/interface.cairo:9
    ○ pool/interface.cairo:16
    ○ pool/interface.cairo:37
    ○ pool/interface.cairo:44
    ○ pool/interface.cairo:49
    ○ pool/interface.cairo:60
    ○ pool/state.cairo:7
    ○ pool/state.cairo:21
    ○ pool/types.cairo:7
    ○ pool/types.cairo:12
    ○ pool/types.cairo:15
    ○ pool/types.cairo:21
    ○ stake/functions.cairo:5
    ○ stake/functions.cairo:5
    ○ stake/functions.cairo:5
    ○ stake/functions.cairo:11
    ○ stake/functions.cairo:12
    ○ stake/functions.cairo:14
    ○ stake/functions.cairo:14
    ○ stake/functions.cairo:16
    ○ stake/functions.cairo:19
```



```
• arithmetic-add Cairo arithmetic is defined over a finite field and has potential for overflows.: 22            –
    ○ administration/functions.cairo:99
    ○ administration/functions.cairo:110
    ○ administration/functions.cairo:126
    ○ administration/state.cairo:64
    ○ administration/state.cairo:65
    ○ administration/state.cairo:87
    ○ administration/types.cairo:22
    ○ bond/functions.cairo:29
    ○ bond/functions.cairo:59
    ○ common/uint64.cairo:26
    ○ common/uint64.cairo:47
    ○ pool/functions.cairo:247
    ○ pool/functions.cairo:248
    ○ pool/state.cairo:72
    ○ pool/state.cairo:73
    ○ pool/state.cairo:94
    ○ pool/state.cairo:121
    ○ pool/types.cairo:176
    ○ pool/types.cairo:176
    ○ stake/functions.cairo:213
    ○ stake/functions.cairo:240
    ○ stake/interface.cairo:281
• must-check-caller-address The function get_caller_address returns 0 when the contract is called directly which might be unexpected: 13            –
    ○ administration/functions.cairo:29
    ○ administration/functions.cairo:49
    ○ bond/functions.cairo:115
    ○ bond/interface.cairo:192
    ○ bond/interface.cairo:250
    ○ pool/functions.cairo:86
    ○ pool/interface.cairo:137
    ○ stake/functions.cairo:34
    ○ stake/functions.cairo:107
    ○ stake/functions.cairo:122
    ○ stake/interface.cairo:169
    ○ stake/interface.cairo:198
    ○ token/interface.cairo:145
```



```
• arithmetic-sub Cairo arithmetic is defined over a finite field and has potential for overflows.: 11            –
    ○ administration/functions.cairo:110
    ○ administration/state.cairo:50
    ○ administration/state.cairo:73
    ○ bond/functions.cairo:187
    ○ common/uint64.cairo:36
    ○ pool/functions.cairo:236
    ○ pool/functions.cairo:278
    ○ pool/state.cairo:55
    ○ pool/state.cairo:80
    ○ pool/state.cairo:121
    ○ stake/interface.cairo:137
• dead-store This variable is assigned or declared here but not used before a return statement.: 11            –
    ○ bond/interface.cairo:141
    ○ common/uint64.cairo:17
    ○ pool/functions.cairo:76
    ○ pool/functions.cairo:77
    ○ pool/functions.cairo:98
    ○ pool/functions.cairo:99
    ○ pool/functions.cairo:111
    ○ pool/functions.cairo:112
    ○ pool/functions.cairo:144
    ○ pool/interface.cairo:96
    ○ stake/interface.cairo:234
• arithmetic-mul Cairo arithmetic is defined over a finite field and has potential for overflows.: 9            –
    ○ administration/types.cairo:23
    ○ bond/functions.cairo:193
    ○ pool/functions.cairo:207
    ○ pool/functions.cairo:207
    ○ pool/functions.cairo:207
    ○ pool/functions.cairo:207
    ○ pool/types.cairo:177
    ○ pool/types.cairo:178
    ○ pool/types.cairo:178
• unused-arguments Unused arguments might indicate a misspelled variable use or unnecessary argument.: 1            –
    ○ pool/types.cairo:61
• unused-function This function is never called.: 1            –
    ○ pool/functions.cairo:122
```

THANK YOU FOR CHOOSING

// HALBORN