Koii Network - K2

L1 Security Assessment

Prepared by: **Halborn**

Date of Engagement: **August 8th, 2023 - December 15th, 2023**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE
0.1	Document Creation	10/12/2023
0.2	Draft Version	10/13/2023
0.3	Draft Review	10/13/2023
1.0	Remediation Plan	10/25/2023
2.0	Document Updates	12/18/2023
2.1	Document Updates	01/05/2024
2.2	Document Updates Review	01/05/2024
2.3	Document Updates	01/20/2024
2.4	Document Updates Review	01/22/2024

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

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Koii Network leverages Solana architecture to enable users to pool their distributed computing power, execute tasks and reward executors and verifiers.

Koii Network engaged Halborn to conduct a security assessment on their L1 network. The security assessment was scoped to the programs provided in the GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

The first assessment was performed between 08/08/2023 and 10/13/2023. In addition, a retest of the task program was done between 11/14/2023 and 12/15/2023.

1.2 ASSESSMENT SUMMARY

The team at Halborn was provided a timeline for the engagement and assigned 1 full-time security engineer to review the security of the code in scope. The security engineer is a blockchain and Solana program security expert with advanced penetration testing and Solana program hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Identify potential security issues within the code

In summary, Halborn identified some security risks that were mostly addressed by the Koii team.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of a manual review of the source code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the assessment. While manual testing is recommended to uncover flaws in business logic, processes, and implementation; automated testing techniques help enhance coverage of the code in scope 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 assessment:

- Research into the architecture, purpose, and use of the platform.
- Manual source code review to identify business logic issues.
- Mapping out possible attack vectors
- Thorough assessment of safety and usage of critical Rust variables and functions in scope that could lead to arithmetic vulnerabilities.
- Finding unsafe Rust code usage (cargo-geiger)
- Scanning dependencies for known vulnerabilities (cargo audit).

2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.

2.1 EXPLOITABILITY

Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

Metrics:

Exploitability Metric (m_E)	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A)	1
	Specific (AO:S)	0.2
Attack Cost (AC)	Low (AC:L)	1
	Medium (AC:M)	0.67
	High (AC:H)	0.33
Attack Complexity (AX)	Low (AX:L)	1
	Medium (AX:M)	0.67
	High (AX:H)	0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

2.2 IMPACT

Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

Metrics:

Impact Metric (m_I)	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact I is calculated using the following formula:

$$I = max(m_I) + \frac{\sum m_I - max(m_I)}{4}$$

2.3 SEVERITY COEFFICIENT

Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient (C)	Coefficient Value	Numerical Value
Reversibility (r)	None (R:N)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope (s)	Changed (S:C)	1.25
	Unchanged (S:U)	1

Severity Coefficient C is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score S is obtained by:

$$S = min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

2.4 SCOPE

Code repositories:

1. K2

- Repository: k2-latest
- Commit ID:
  - Initial: 2bc3e0a
  - Retest: 3469b3c81
- Files and Directories In Scope:
  - k2-latest/clap-utils/src/compute_unit_price.rs
  - k2-latest/clap-utils/src/input_validators.rs
  - k2-latest/clap-utils/src/keypair.rs
  - k2-latest/clap-v3-utils/src/input_validators.rs
  - k2-latest/clap-v3-utils/src/keypair.rs
  - k2-latest/faucet/src/bin/faucet.rs
  - k2-latest/faucet/src/faucet.rs
  - k2-latest/genesis/src/main.rs
  - k2-latest/install/src/command.rs
  - k2-latest/install/src/defaults.rs
  - k2-latest/install/src/lib.rs
  - k2-latest/keygen/src/keygen.rs
  - k2-latest/ledger/src/blockstore_processor.rs
  - k2-latest/ledger-tool/src/main.rs
  - k2-latest/program-runtime/src/pre_account.rs
  - k2-latest/program-runtime/src/prioritization_fee.rs
  - k2-latest/programs/address-lookup-table/src/processor.rs
  - k2-latest/programs/attention/*
  - k2-latest/programs/bpf/c/src/dup_accounts/dup_accounts.c
  - k2-latest/programs/bpf/rust/dup_accounts/src/lib.rs
  - k2-latest/programs/bpf/rust/get_minimum_delegation/src/lib.rs
  - k2-latest/programs/bpf/rust/invoke/src/processor.rs
  - k2-latest/programs/stake/src/stake_state.rs
  - k2-latest/programs/task-program/*

- k2-latest/runtime/src/account_rent_state.rs
- k2-latest/runtime/src/accounts_db.rs
- k2-latest/runtime/src/bank.rs
- k2-latest/runtime/src/builtins.rs
- k2-latest/runtime/src/nonce_keyed_account.rs
- k2-latest/runtime/src/system_instruction_processor.rs
- k2-latest/sdk/program/src/attention
- k2-latest/sdk/program/src/clock.rs
- k2-latest/sdk/program/src/instruction.rs
- k2-latest/sdk/program/src/lib.rs
- k2-latest/sdk/program/src/program_error.rs
- k2-latest/sdk/program/src/rent.rs
- k2-latest/sdk/program/src/system_instruction.rs
- k2-latest/sdk/program/src/task_program
- k2-latest/sdk/src/feature_set.rs
- k2-latest/sdk/src/genesis_config.rs
- k2-latest/sdk/src/transaction_context.rs
- k2-latest/stake-accounts/src/arg_parser.rs
- k2-latest/stake-accounts/src/main.rs
- k2-latest/tokens/src/arg_parser.rs
- k2-latest/transaction-dos/src/main.rs
- k2-latest/validator/src/bin/solana-test-validator.rs
- k2-latest/validator/src/bootstrap.rs
- k2-latest/watchtower/src/main.rs

Please note, the code in the k2-latest repository is a fork of Solana with a number of changes introduced. Above are listed the files in the scope of this assessment. The reference Solana implementation was considered stable for the purposes of this security review.

Out-of-scope:
- third-party libraries and dependencies
- financial-related attacks

3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
3	0	2	5	1

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) INTEGER OVERFLOW LEADS TO UNFAIR REWARD DISTRIBUTION	Critical (10)	SOLVED - 12/22/2023
(HAL-02) ANONYMOUS WRITE ACCESS TO TASKS	Critical (10)	SOLVED - 08/18/2023
(HAL-03) HIJACKING STAKE POT ACCOUNTS	Critical (10)	SOLVED - 08/23/2023
(HAL-04) SORTING METHOD LEADS TO UNFAIR REWARD DISTRIBUTION	Medium (6.7)	SOLVED - 12/22/2023
(HAL-05) FUNDING AND DELETING INCORRECT STAKE POT ACCOUNTS	Medium (5.0)	SOLVED - 08/31/2023
(HAL-06) HARDCODED OWNER ACCOUNT ADDRESSES	Low (3.5)	SOLVED - 05/09/2023
(HAL-07) VANITY ADDRESSES REQUIRED FOR STAKE POT ACCOUNTS	Low (2.5)	NOT APPLICABLE
(HAL-08) PRINTING CONFIDENTIAL DATA TO CONSOLE	Low (2.5)	SOLVED - 08/18/2023
(HAL-09) ACCOUNT TYPE CHECK MISSING LEADS TO DATA CORRUPTION	Low (2.0)	SOLVED - 08/22/2023
(HAL-10) USERS CAN BE FORCED TO FUND THE MANAGERS ACCOUNT	Low (2.0)	SOLVED - 12/22/2023
(HAL-11) IP ADDRESSES POTENTIALLY STORED ON CHAIN	Informational (1.7)	ACKNOWLEDGED

FINDINGS & TECH DETAILS

4.1 (HAL-01) INTEGER OVERFLOW LEADS TO UNFAIR REWARD DISTRIBUTION - CRITICAL(10)

Description:

The attention program enables users to create and vote on reward distribution lists. Rewards are distributed every epoch and the runtime calculates the amounts based on the data stored in the distribution_data account.

Reward amounts are defined in signed 64-bit integers, so recipient accounts can be debited or credited every epoch. When a reward amount is negative, the runtime casts it to u64 to ensure the new amount is greater than or equal to zero. When the new amount is greater than the current recipient account balance, the balance is cleared. When it is not, the recipient is credited the new amount.

This design assumes casting a i64 variable to u64 as functionally equivalent to taking the absolute value of the i64 variable value, however in reality, the Rust programming language calculates the 2-complement of the negative value, resulting in reward amounts significantly different from expected.

An example effect of the above is when an account is debited a low amount, the casting operation inflates disproportionately the supposed reward and results in the runtime clearing that account's balance unfairly.

What is more, if an account does have its balance greater than the cast amount, it is credited that amount which in certain cases may underflow and inflate the balance of the penalized account.

Please note, this issue was also identified in the task program.

Code Location:

Listing 1: k2-latest/runtime/src/bank.rs (Lines 3393,3397)

```
3392 } else if distribution.1.is_negative() {
3393     if recipient_account.lamports() < distribution.1 as u64 {
3394         recipient_account.set_lamports(0);
3395     } else {
3396         recipient_account
3397             .set_lamports(recipient_account.lamports() +
   ↳ distribution.1 as u64);
3398     }
3399 }
3400 self.store_account(
3401     &Pubkey::from_str(&distribution.0).unwrap_or_default(),
3402     &recipient_account,
3403 );
```

Listing 2: k2-latest/programs/task-program/src/task_instruction.rs (Lines 2003,2006)

```
1999 } else if distribution.1.is_negative() {
2000     if task.stake_list.get_mut(&distribution.0).is_none() {
2001         continue;
2002     }
2003     if *task.stake_list.get(&distribution.0).unwrap() <
   ↳ distribution.1 as u64 {
2004         *task.stake_list.get_mut(&distribution.0).unwrap() = 0;
2005     } else {
2006         *task.stake_list.get_mut(&distribution.0).unwrap() -=
   ↳ distribution.1 as u64;
2007     }
2008 }
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:C/R:N/S:U (10)

Recommendation:

Instead of casting i64 to u64 in line 3393 in the bank.rs file and in line 2003 in the task_instruction file, take the absolute value of the reward with the .abs() method. In lines 3397 and 2006 respectively, also instead of casting take the absolute value of the reward, and debit the account this absolute value.

Remediation Plan:

SOLVED: The Koii Network team solved this issue in commit 4dd2fe6.

4.2 (HAL-02) ANONYMOUS WRITE ACCESS TO TASKS - CRITICAL(10)

Description:

The task program enables users to create and manage off chain tasks. As the CreateTask instruction handler is missing protection against account overwriting, it is possible that anyone can overwrite an existing TaskState account and delete any task in any state and which results in breaking the entire task processing flow.

What is more, due to this attack, stake allocated in stake_pot_account is locked. It's not possible to assign the same stake_pot_account to another task to withdraw tokens.

Additionally, because the task program is authorized by the runtime to debit any account, due to the same missing signer check, an attacker can reference another user as the task manager, making the program charge the user for the creation of the task.

Code Location:

Listing 3: k2-latest/programs/task-program/src/task_instruction.rs (Line 665)

```
661 let transaction_context = &invoke_context.transaction_context;
662 let instruction_context = transaction_context.
  ↳ get_current_instruction_context()?;
663
664 // let account_info_iter = &mut accounts.iter();
665 let mut manager_info =
666     instruction_context.try_borrow_instruction_account(
  ↳ transaction_context, 0)?;
```

Listing 4: k2-latest/programs/task-program/src/task_instruction.rs (Line 727)

```
711 if manager_info.get_lamports() < total_bounty_amount {
712     return Err(InstructionError::InsufficientFunds);
713 }
714 manager_info.checked_sub_lamports(total_bounty_amount)?;
715 stake_pot_account.checked_add_lamports(total_bounty_amount)?;
716
717 if round_time < (audit_window + submission_window) {
718     return Err(InstructionError::BorshIoError(
719         "round_time must be greater than audit_window +
 ↳ submission_window".to_string(),
720     ));
721 }
722 check_account_owner(&task_state_info, program_id)?;
723 check_account_owner(&stake_pot_account, program_id)?;
724
725 let task = TaskState {
726     task_name: String::from_utf8_lossy(&task_name).trim().
 ↳ to_string(),
727     task_manager: *manager_info.get_key(),
728     task_audit_program: String::from_utf8_lossy(&
 ↳ task_audit_program)
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:C/A:N/D:N/Y:N/R:N/S:U (10)

Recommendation:

It is recommended to introduce a check to verify if an account is already initialized to prevent re-initialization and check if the manager signed the transaction with the CreateTask instruction.

Remediation Plan:

SOLVED: The Koii Network team solved this issue in commit bd3eae7 by adding the manager signer check and verifying if the account to be created has not been initialized already.

4.3 (HAL-03) HIJACKING STAKE POT ACCOUNTS - CRITICAL(10)

Description:

The task program enables users to create and manage off chain tasks. The UpdateTask instruction allows the task manager to migrate task and stake pot data to another account. Because the UpdateTask instruction handler fails to check if the new account has not been initialized already, any task owner can overwrite any task data and hijack stake pots, becoming the new manager for both.

Code Location:

Listing 5: k2-latest/programs/task-program/src/task_instruction.rs (Line 665)

```rust
if !new_stake_pot_account
    .get_key()
    .to_string()
    .starts_with("stakepotaccount")
{
    return Err(InstructionError::BorshIoError(
        "New stake_pot_account publickey must start with
 ↳ stakepotaccount".to_string(),
    ));
}
if !stake_pot_account
    .get_key()
    .to_string()
    .starts_with("stakepotaccount")
{
    return Err(InstructionError::BorshIoError(
        "stake_pot_account public key must start with
 ↳ stakepotaccount".to_string(),
    ));
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:C/A:N/D:N/Y:N/R:N/S:U (10)

Recommendation:

Prevent the task manager from updating their task if the new task account has already been initialized.

Remediation Plan:

SOLVED: The Koii Network team solved this issue in commit f4fe744 by requiring the sender to provide the new task account signature and reverting the transaction if the new task account is already initialized.

4.4 (HAL-04) SORTING METHOD LEADS TO UNFAIR REWARD DISTRIBUTION - MEDIUM (6.7)

Description:

The attention program enables users to create and vote on reward distribution lists. Rewards are distributed every epoch and the runtime calculates the amounts based on the data stored in the distribution_data account.

Network participants can create and submit their distribution lists for other users to review and audit if necessary. When the voting period is over, all submitted lists are processed by the runtime, which chooses a list that was either audited successfully or wasn't audited at all.

The runtime aggregates audit results in a BTreeMap structure where the submitter addresses are keys are audit results are values. The runtime then selects the distribution list to process by creating an Iterator from the structure and getting the last element in that iterator.

An Iterator created from a BTreeMap is sorted by key, which means the selection ultimately depends not on the total number of votes nor on the time of submission but the submitter public key. This can be exploited by generating a vanity address that would always appear last in the Iterator to ensure a high probability of being selected.

Please note, this issue was also identified in the task program.

Code Location:

Listing 6: k2-latest/runtime/src/bank.rs (Line 3148)

```
3144 if score_sum.iter().next_back().is_none() {
3145     msg!("Both distribution have audits but no votes to decide,
  ↳ Payout cannot proceed");
3146     return;
3147 }
3148 let selected = score_sum.iter().next_back().unwrap();
3149
3150 if score_sum.len() > 1 {
3151     // Verifying the second selected don't have the same score
3152     let mut temp = score_sum.clone();
3153     temp.remove(selected.0);
3154     if !temp.iter().next_back().is_none() {
3155         let selected_temp = temp.iter().next_back().unwrap();
3156         if selected.1 == selected_temp.1 {
3157             msg!("Both distribution have same number of votes on
  ↳ audits, Payout cannot proceed");
3158             return;
3159         }
3160     }
3161 }
```

Listing 7: task-program/src/task_instruction.rs (Line 1723)

```
1716 if score_sum.iter().next_back().is_none() {
1717     msg!("Both distribution have audits but no votes to decide,
  ↳ Payout cannot proceed");
1718     return Err(InstructionError::BorshIoError(
1719         "Both distribution have audits but no votes to decide,
  ↳ Payout cannot proceed"
1720             .to_string(),
1721     ));
1722 }
1723 let selected = score_sum.iter().next_back().unwrap();
1724
1725 if score_sum.len() > 1 {
1726     // Verifying the second selected don't have the same score
1727     let mut temp = score_sum.clone();
1728     temp.remove(selected.0);
1729     if !temp.iter().next_back().is_none() {
1730         let selected_temp = temp.iter().next_back().unwrap();
```

```
        if selected.1 == selected_temp.1 {
            return Err(InstructionError::BorshIoError(
            "Both distribution have same number of votes on audits
 ↳ , Payout cannot proceed"
                .to_string(),
        ));
        }
    }
}
```

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:N/D:N/Y:C/R:N/S:U (6.7)

Recommendation:

Reconsider the rules for selecting the distribution list: possible solutions include sorting by the number of votes or total score.

Remediation Plan:

SOLVED: The issue was solved by the Koii Network team.

4.5 (HAL-05) FUNDING AND DELETING INCORRECT STAKE POT ACCOUNTS - MEDIUM (5.0)

Description:

With the FundTask instruction, the submitter deposits their funds in a stake pot account. The program allows the submitter to fund any stake_pot they like, including stake_pots that are different from the one assigned to the task that he/she intended to fund. In the result, amount of roe in the stake_pot account can differ from parameters assigned to task_state account (total_bounty_amount) which may have unexpected consequences on task bounty payout as stake_pot account would have less balance than expected.

Similarly, the DeleteTask instruction users can send when they want to decommission their tasks does verify if the provided stake_pot account matches the address in the task_state they are about to delete.

Code Location:

Listing 8: k2-latest/runtime/src/bank.rs (Line 3148)

```
3144 if score_sum.iter().next_back().is_none() {
3145     msg!("Both distribution have audits but no votes to decide,
     ↳ Payout cannot proceed");
3146     return;
3147 }
3148 let selected = score_sum.iter().next_back().unwrap();
3149
3150 if score_sum.len() > 1 {
3151     // Verifying the second selected don't have the same score
3152     let mut temp = score_sum.clone();
3153     temp.remove(selected.0);
3154     if !temp.iter().next_back().is_none() {
3155         let selected_temp = temp.iter().next_back().unwrap();
3156         if selected.1 == selected_temp.1 {
3157             msg!("Both distribution have same number of votes on
↳ audits, Payout cannot proceed");
                return;
            }
        }
    }
```

BVSS:

AO:A/AC:L/AX:M/C:N/I:N/A:N/D:H/Y:N/R:N/S:U (5.0)

Recommendation:

Consider verifying if the funded stake_pot account is assigned to the task owned by the transaction sender.

Remediation Plan:

SOLVED: The Koii Network team solved this issue in commit f019c8b by checking if the funded stake_pot account is assigned to the task owned by the submitter.

4.6 (HAL-06) HARDCODED OWNER ACCOUNT ADDRESSES - LOW (3.5)

Description:

Several features of the task program are restricted to program owners only. Those addresses are hardcoded in the top-level program instruction handler and cannot be updated in any way, which may be an issue in case an owner starts acting maliciously, or their account is compromised because removing an owner requires a hard fork of the network.

Code Location:

Listing 9: k2-latest/programs/task-program/src/task_instruction.rs (Lines 31-36)

```
26 pub fn process_instruction(
27     _first_instruction_account: usize,
28     invoke_context: &mut InvokeContext,
29 ) -> Result<(), InstructionError> {
30     let owner_pubkeys = vec![
31         "TaskMDbVartWDkgHrWZ6NiHUQUwdWpN3WDyRmSEoMTm".to_string(),
32         "J9mdajjDZcT13U3aq3cEVH8WwZaMfhfp76DTeWNUmLZL".to_string(),
33         "7Jo84gCrqv5fKYVAD3YHsanYmq4THF8Q7dV6RbiRLp5M".to_string(),
34         "4ryyYHVqpi7XHu366J2ifTtE3fk6mmdR1z7ar43vWJxU".to_string(),
35         "HLPcVSUTGHJXMRp1SXYsVyxu6ho2VXQkc1cbj6wZZ4hV".to_string(),
36         "FnQm11NXJxPSjza3fuhuQ6Cu4fKNqdaPkVSRyLSWf14d".to_string(),
37     ];
38     let transaction_context = &invoke_context.transaction_context;
39     let instruction_context = transaction_context.get_current_instruction_context()?;
```

BVSS:

AO:A/AC:L/AX:H/C:N/I:C/A:L/D:N/Y:N/R:N/S:U (3.5)

Recommendation:

Consider creating a Config account that would store program owner addresses and implement an instruction which allows owners to modify that config.

Remediation Plan:

SOLVED: The Koii Network team solved this issue in commit 3469b3c by deprecating the hardcoded owner addresses and introducing program owner list management functionality to the Task program.

4.7 (HAL-07) VANITY ADDRESSES REQUIRED FOR STAKE POT ACCOUNTS - LOW (2.5)

Description:

In the CreateTask instruction handler in the task program, it is required that the provided stake_pot account is unique for each task. However, there is also a requirement that the address of this account should start with the stakepotaccount string. It is not an easy task to generate such an address, and generating it can take a couple of hours (using the solana grind command). Which results in a very inefficient system and works against scalability.

Code Location:

Listing 10: k2-latest/programs/task-program/src/task_instruction.rs (Lines 285,294)

```
if !new_stake_pot_account
    .get_key()
    .to_string()
    .starts_with("stakepotaccount")
{
    return Err(InstructionError::BorshIoError(
        "New stake_pot_account publickey must start with
↳ stakepotaccount".to_string(),
    ));
}
if !stake_pot_account
    .get_key()
    .to_string()
    .starts_with("stakepotaccount")
{
    return Err(InstructionError::BorshIoError(
        "stake_pot_account public key must start with
↳ stakepotaccount".to_string(),
    ));
}
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:L/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

Consider using PDAs instead of vanity addresses.

Remediation Plan:

NOT APPLICABLE: The Koii Network team believes this issue is not applicable to their current process:

> we are not grinding the private keys that start with the stakepotaccount.

4.8 (HAL-08) PRINTING CONFIDENTIAL DATA TO CONSOLE - LOW (2.5)

Description:

The Koii Network team modified the Solana code responsible for generating private keys, and now it prints the seed phrase and the generated key to the console. While this can be considered convenient by users, it is against existing good practice, especially when using shared devices, allowing for shoulder surfing types of attacks.

Code Location:

Listing 11: k2-latest/clap-utils/src/keypair.rs

```rust
let mnemonic = parse_language_fn()?;
let seed = Seed::new(&mnemonic, &passphrase);
msg!(
    "mnemonic {:?} passphrase {:?} seed {:?} legacy {:?}",
    mnemonic,
    passphrase,
    seed,
    legacy
);
```

BVSS:

AO:A/AC:L/AX:L/C:N/I:N/A:L/D:N/Y:N/R:N/S:U (2.5)

Recommendation:

Instead of printing the secrets to stdout, consider following the best practices established by ssh-keygen and output the mnemonic phrase and the private key to a file readable only by the user in whose context the binary is executed in.

Remediation Plan:

SOLVED: The Koii Network team fixed this issue in commit e004c9d by removing the log message from the code.

4.9 (HAL-09) ACCOUNT TYPE CHECK MISSING LEADS TO DATA CORRUPTION - LOW (2.0)

Description:

Along with the attention and task programs, in K2 the Koii Network team introduces several new types of accounts, designed to persist data essential to facilitate reward payouts.

In the KoiiVarManager instruction handler, before parsing the provided accounts the program verifies if there has been enough space allocated and if it has, it attempts to parse the account data to the expected format, using the unwrap_or_default() method to obtain the desired data structure.

The unwrap_or_default function tries to extract the value from the Option or Result if it is Ok or Some respectively, and if it fails to do that it creates the default value of the type. Because the programs do not check if the provided account type matches the expected type, owners can overwrite or zero out any account owned by the task program, corrupting a state essential to pay out rewards to network participants.

Code Location:

Listing 12: k2-latest/programs/task-program/src/task_instruction.rs (Line 538)

```rust
let data = koii_vars_account.get_data().to_vec();
let koii_vars_result: Result<BTreeMap<String, String>, Error> =
 ↳ serde_json::from_slice(&data);
let mut koii_vars;
if koii_vars_result.is_ok() {
    koii_vars = koii_vars_result.unwrap()
} else {
    koii_vars = koii_vars_result.unwrap_or_default()
    // return Err(InstructionError::InvalidAccountData);
}
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:C/A:N/D:N/Y:N/R:N/S:U (2.0)

Recommendation:

Consider reverting the transaction if the provided koii_vars_account cannot be deserialized to the desired type.

Remediation Plan:

SOLVED: The Koii Network team solved this issue in commit b2d1726 by reverting the transaction if the provided koii_vars_account is not empty, and it does not deserialize correctly.

4.10 (HAL-10) USERS CAN BE FORCED TO FUND THE MANAGERS ACCOUNT - LOW (2.0)

Description:

The task program enables users to create and manage off chain tasks. Within the program, there are some privileged manager users, a registry of whom is maintained in the Managers account. Managers can be added and removed, and the initial set is created with the HandleManagerAccounts/init instruction/operation. On initialization, this account is credited 100 KOII. Because the Koii team modified the Solana runtime execution rules such that the task program can debit any account and there is no signer check, theoretically any account in Koii can be provided as owner_signer and serve as the source of liquidity for the Managers account.

Code Location:

Listing 13: k2-latest/programs/task-program/src/task_instruction.rs (Line 2443)

```
2432 if operation == "init" {
2433     let is_initialized = manager_account_data_holder
2434         .get_data()
2435         .iter()
2436         .any(|&byte| byte != 0);
2437     if is_initialized {
2438         return Err(InstructionError::AccountAlreadyInitialized);
2439     }
2440     let mut owner_signer =
2441         instruction_context.try_borrow_instruction_account(
  ↳ transaction_context, 1)?;
2442     let amount = 100_000_000_000;
2443     owner_signer.checked_sub_lamports(amount)?;
2444     manager_account_data_holder.checked_add_lamports(amount)?;
2445     manager_account_data_holder.set_owner(&id().to_bytes())?;
2446     let manager_pubkeys = vec![
2447         "TaskMDbVartWDkgHrWZ6NiHUQUwdWpN3WDyRmSEoMTm".to_string(),
        "J9mdajjDZcT13U3aq3cEVH8WwZaMfhfp76DTeWNUmLZL".to_string()
    ↳ ,
        "7Jo84gCrqv5fKYVAD3YHsanYmq4THF8Q7dV6RbiRLp5M".to_string()
    ↳ ,
        "4ryyYHVqpi7XHu366J2ifTtE3fk6mmdR1z7ar43vWJxU".to_string()
    ↳ ,
        "HLPcVSUTGHJXMRp1SXYsVyxu6ho2VXQkc1cbj6wZZ4hV".to_string()
    ↳ ,
        "FnQm11NXJxPSjza3fuhuQ6Cu4fKNqdaPkVSRyLSWf14d".to_string()
    ↳ ,
    ];
    let serialized_json_data = serde_json::to_vec(&manager_pubkeys
    ↳ ).unwrap_or_default();
    manager_account_data_holder.set_data(&serialized_json_data)?;
}
```

Listing 14: k2-latest/program-runtime/src/pre_account.rs

```
if (!task_program::program::check_id(program_id) && program_id !=
↳ pre.owner()) // line coverage used to get branch coverage
     && pre.lamports() > post.lamports()
    {
        return Err(InstructionError::ExternalAccountLamportSpend);
    }
```

BVSS:

AO:S/AC:L/AX:L/C:N/I:N/A:N/D:C/Y:N/R:N/S:U (2.0)

Recommendation:

Consider verifying if the owner_signer signed the init operation.

Remediation Plan:

SOLVED: The Koii Network team solved this issue in commit 7035e1e.

4.11 (HAL-11) IP ADDRESSES POTENTIALLY STORED ON CHAIN - INFORMATIONAL (1.7)

Description:

The task program enables users to create and manage off chain tasks. A task record, among other parameters, stores a list of IP address of all users who staked their KOII in the task program. Because all on-chain data is public, exposing users' net worth and potentially mapping it to their geographical location directly increases the risk of a targeted cybercrime or a physical assault.

Code Location:

Listing 15: k2-latest/programs/task-program/src/task_instruction.rs (Line 920)

```
917 let ip_address_str = String::from_utf8_lossy(&ip_address).trim().to_string();
918 if ip_address_str != "" {
919     task.ip_address_list
920         .insert(submitter_info.get_key().to_string(), ip_address_str);
921 }
922 msg!("TASK STATE {:?}", task);
923 let serialized_json_data = serde_json::to_vec(&task).unwrap_or_default();
924 task_state_info.set_data(&serialized_json_data)?;
```

BVSS:

AO:A/AC:H/AX:H/C:N/I:N/A:N/D:C/Y:C/R:N/S:C (1.7)

Recommendation:

Do not collect users’ IP addresses.

Remediation Plan:

ACKNOWLEDGED: The Koii Network team acknowledged this finding. They commented:

> Since the nodes runs tasks and some of these tasks may require the node to be accessible over the internet, that’s why we have the IP Addresses field. However, we don’t store direct IP addresses, we store domain names here. For the domain there are 2 types available, One is the UPnP which will resolve directly to the node’s IP address however the other one is subdomain and the requests are relayed from an external server. So, anyone opting for more security can use that option instead of directly exposing the hostname resolving to his IP address.

> IP addresses are not required to be stored on chain, and we recommend that CNAMES are used here with appropriate security precautions. Even so, DNS posts are only required for certain nodes, which aids the discovery process.

> IP posting is not required by default, and only used via CNAMEs for some specific task roles

AUTOMATED TESTING

5.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with the detection of well-known security issues and vulnerabilities. Among the tools used was cargo-audit, a security scanner for vulnerabilities reported to the Rust-Sec Advisory Database. All vulnerabilities published in https://crates.io are stored in a repository named The RustSec Advisory Database. cargo audit is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the reviewers are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

Results:

k2-latest/Cargo.lock

ID	package	Short Description
RUSTSEC-2022-0093	ed25519-dalek	Double Public Key Signing Function Oracle Attack on ‘ed25519-dalek‘
RUSTSEC-2023-0063	quinn-proto	Denial of service in Quinn servers
RUSTSEC-2023-0001	tokio	reject_remote_clients Configuration corruption
RUSTSEC-2023-0065	tungstenite	Tungstenite allows remote attackers to cause a denial of service
RUSTSEC-2023-0052	webpki	webpki: CPU denial of service in certificate path building

k2-latest/programs/bpf/Cargo.lock

ID	package	Short Description
RUSTSEC-2022-0093	ed25519-dalek	Double Public Key Signing Function Oracle Attack on ‘ed25519-dalek‘
RUSTSEC-2023-0063	quinn-proto	Denial of service in Quinn servers
RUSTSEC-2023-0001	tokio	reject_remote_clients Configuration corruption
RUSTSEC-2023-0065	tungstenite	Tungstenite allows remote attackers to cause a denial of service
RUSTSEC-2023-0052	webpki	webpki: CPU denial of service in certificate path building
RUSTSEC-2020-0159	chrono	Potential segfault in ‘localtime_r‘ invocations
RUSTSEC-2022-0093	ed25519-dalek	Double Public Key Signing Function Oracle Attack on ‘ed25519-dalek‘
RUSTSEC-2023-0044	openssl	‘openssl‘ ‘X509VerifyParamRef::set_host‘ buffer over-read
RUSTSEC-2023-0063	quinn-proto	Denial of service in Quinn servers
RUSTSEC-2020-0071	time	Potential segfault in the time crate
RUSTSEC-2023-0001	tokio	reject_remote_clients Configuration corruption
RUSTSEC-2023-0065	tungstenite	Tungstenite allows remote attackers to cause a denial of service
RUSTSEC-2023-0052	webpki	webpki: CPU denial of service in certificate path building
RUSTSEC-2023-0052	webpki	webpki: CPU denial of service in certificate path building

5.2 UNSAFE RUST CODE DETECTION

Description:

Halborn used automated security scanners to assist with the detection of well-known security issues and vulnerabilities. Among the tools used was cargo-geiger, a security tool that lists statistics related to the usage of unsafe Rust code in a core Rust codebase and all its dependencies.

Results:

No new crates containing unsafe Rust code were identified.

THANK YOU FOR CHOOSING

// HALBORN