HALBORN

Phantasia Sports

Solana Program Security Audit

Prepared by: **Halborn**

Date of Engagement: **November 22nd, 2021 - December 2nd, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	11/22/2021	Piotr Cielas
0.2	Document Edits	11/24/2021	Piotr Cielas
0.3	Document Edits	11/25/2021	Piotr Cielas
0.4	Final Draft	11/29/2021	Piotr Cielas
0.5	Draft Review	12/02/2021	Gabi Urrutia
1.0	Remediation Plan	12/28/2021	Piotr Cielas
1.1	Remediation Plan Review	01/07/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Phantasia is a Fantasy Sports platform that is built on top of the Solana Blockchain. On Phantasia, anyone can earn tokens by contributing to the ecosystem through skilled gameplay. Players can join public or private tournaments to play fantasy sports against other users.

Phantasia engaged Halborn to conduct a security assessment on their gaming program beginning on November 22nd, 2021 and ending December 2nd, 2021. This security assessment was scoped to the phantasia-solana-program repository and an audit of the security risk and implications regarding the changes introduced by the development team at Phantasia prior to its production release shortly following the assessment's deadline.

1.2 AUDIT SUMMARY

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

The purpose of this audit to achieve the following:

- Ensure that program functions are intended.
- Identify potential security issues with the program.

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

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual view of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the program audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of programs 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, purpose, and use of the platform.
- Manual code read and walkthrough.
- Manual Assessment of use and safety for the critical Rust variables and functions in scope to identify any arithmetic related vulnerability classes.
- Fuzz testing. (Halborn custom fuzzing tool).
- Checking the test coverage. (cargo tarpaulin)
- Scanning of Rust files for vulnerabilities.(cargo audit).

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

This review was scoped to the main Solana program.

1. Solana program
   (a) Repository: phantasia-solana-program
   (b) Commit ID: d8c8c7f7276d02d33b4bb0723ab1a72c68b5cc2f

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	2	2	3	3

LIKELIHOOD

IMPACT

🟨	🟨	🟥	🟥	🟪
🟩	🟨 (HAL-04)	🟨	🟥	🟥
🟩 (HAL-05) (HAL-06) (HAL-07)	🟩	🟨 (HAL-03)	🟨	🟥 (HAL-01) (HAL-02)
⬜	🟩	🟩	🟨	🟨
⬜ (HAL-08) (HAL-09) (HAL-10)	⬜	🟩	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) ANONYMOUS FAN CARD MODIFICATION	High	SOLVED - 11/26/2021
(HAL-02) CONTEST DOS	High	SOLVED - 01/06/2022
(HAL-03) HARDCODED GOVERNANCE ADDRESS	Medium	ACKNOWLEDGED
(HAL-04) SALARY CAP CONTEST PARAMETER VALIDATION MISSING	Medium	ACKNOWLEDGED
(HAL-05) LEAGUE/GAME TYPE CONTEST PARAMETERS VALIDATION MISSING	Low	SOLVED - 12/13/2021
(HAL-06) INTEGER OVERFLOW	Low	SOLVED - 12/13/2021
(HAL-07) INDEX OUT OF BOUNDS	Low	SOLVED - 12/13/2021
(HAL-08) POSSIBLE MISUSE OF HELPER METHODS	Informational	SOLVED - 01/06/2022
(HAL-09) CONTEST CREATOR VALIDATION MISSING	Informational	SOLVED - 12/13/2021
(HAL-10) CREATING FAN CARDS FROM FROZEN NFTS ALLOWED	Informational	SOLVED - 12/13/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) ANONYMOUS FAN CARD MODIFICATION - HIGH

Description:

Users can create and update fan cards with the UpdateOrCreateFanCard instruction. Along with the instruction data they have to provide several accounts, one of them being fan_card_storage_account. This account stores a FanCard struct. The struct is defined in util.rs and has multiple fields, including user_main: Pubkey. This field denotes the card creator.

The UpdateOrCreateFanCard instruction handler does not validate the transaction signer account address to match user_main of the provided fan_card_storage_account and in consequence a malicious user may over-write **all** fan cards.

Code Location:

Listing 1: processor/update_or_create_fan_card_v1.rs (Lines 23,25,29)

```
19 pub fn process_update_or_create_fan_card_v1(
20     accounts: &[AccountInfo],
21 ) -> ProgramResult {
22     let account_info_iter = &mut accounts.iter();
23     let user_main_account = next_account_info(account_info_iter)?;
24 
25     if !user_main_account.is_signer {
26         return Err(ProgramError::MissingRequiredSignature);
27     }
28 
29     let fan_card_storage_account = next_account_info(
           account_info_iter)?;
```

Listing 2: processor/update_or_create_fan_card_v1.rs (Lines 125)

```
103 let mut fan_card_data_byte_array = fan_card_storage_account.data.try_borrow_mut().unwrap();
104     if fan_card_data_byte_array[0] == 0u8 {
105         let rent = Rent::get()?;
106 
107         if !rent.is_exempt(fan_card_account_lamports, fan_card_data_bytes) {
108             msg!("PhantasiaError::NotRentExempt");
109             return Err(PhantasiaError::NotRentExempt.into());
110         }
111 
112         if fan_card_data_bytes != state::FAN_CARD_STORAGE_TOTAL_BYTES {
113             msg!("PhantasiaError::DataSizeNotMatched");
114             return Err(PhantasiaError::DataSizeNotMatched.into());
115         }
116 
117         fan_card_data_byte_array[0] = AccTypesWithVersion::FanCardDataV1 as u8;
118     } else if fan_card_data_byte_array[0] != AccTypesWithVersion::FanCardDataV1 as u8 {
119         msg!("PhantasiaError::ExpectedAccountTypeMismatched");
120         return Err(PhantasiaError::ExpectedAccountTypeMismatched.into());
121     }
122 
123 
124     let user_fan_card: FanCard = FanCard {
125         user_main: *user_main_account.key,
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendations:

Validate the user_main_account key to match user_main on card update.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit 5c83e22e08086abad97119860485316c013ae748. The instruction handler validates if the transaction signer is the owner of the fan card.

3.2 (HAL-02) CONTEST DOS - HIGH

Description:

Users can join contests by sending the AddOrUpdateParticipantData instruction. The instruction handler however does not verify if the user has already submitted a roster which combined with other vulnerabilities identified in the program allows a single attacker to fill up any and all contests with bogus or duplicated entries thus preventing legitimate users from entering contests.

An attack of this type doesn't carry any immediate loss of funds risk however it would be extremely harmful to the protocol to be unable to carry out normal operations.

Code Location:

Listing 3: processor/add_or_update_participant_data.rs (Lines 20,22)

```
14 pub fn process_add_or_update_participant_data(
15     accounts: &[AccountInfo],
16     roster_9_players_input: [u32; 9],
17     participant_index_input: u16,
18 ) -> ProgramResult {
19     let account_info_iter = &mut accounts.iter();
20     let participant_main_account = next_account_info(
           account_info_iter)?;
21 
22     if !participant_main_account.is_signer {
23         msg!("ProgramError::MissingRequiredSignature");
24         return Err(ProgramError::MissingRequiredSignature);
25     }
```

Listing 4: processor/add_or_update_participant_data.rs (Lines 71,88)

```
70 let participant_data = ParticipantDataV1 {
71     participant_main_pkey: *participant_main_account.key,
72     roster_9_players: roster_9_players_input,
73     last_updated_timestamp: now,
};
state::pack_to_participant_v1(
    participant_data,
    &mut contest_data_byte_array[participant_data_si..
        participant_data_ei],
)
.unwrap();
} else {
let participant_data_si = state::CONTEST_INFO_HEADER_V1_DATA_BYTES
    + (participant_index_input as usize) * state::
        PARTICIPANT_V1_DATA_BYTES;
let participant_data_ei = participant_data_si + state::
    PARTICIPANT_V1_DATA_BYTES;
let mut participant_data = state::unpack_to_participant_v1(
    &contest_data_byte_array[participant_data_si..
        participant_data_ei],
)
.unwrap();
if participant_data.participant_main_pkey != *
    participant_main_account.key {
    msg!("PhantasiaError::WrongParticipantIndex");
    return Err(PhantasiaError::WrongParticipantIndex.into());
}
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendations:

Restrict the number of rosters per participant_main_account to one.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit d19c0f12f26ddce1adc605ce19ca24a085acefb6. The number of rosters per participant_main_account is restricted to one now.

3.3 (HAL-03) HARDCODED GOVERNANCE ADDRESS - MEDIUM

Description:

The ClaimContestPrizes, CreateWinnersAccountV1, CloseWinnersAccount and TransferNFTtoWinner instruction handlers use the get_admin_pubkey() utility function to validate the signer address. This function returns a hardcoded governance account address. Since this address is hardcoded it cannot be modified without redeploying the program if the account compromised.

Code Location:

Listing 5: utils.rs (Lines 13)

```
pub fn get_admin_pubkey() -> Pubkey {
    let admin_pubkey_str: &'static str =
        env!("ADMIN_PUBKEY", "Must specify a admin account public
            key!");
    msg!(
        "the ADMIN_PUBKEY variable at the time of compiling was:
            {}",
        admin_pubkey_str
    );
    let pubkey_vec = bs58::decode(admin_pubkey_str).into_vec().
        unwrap();
    let admin_pubkey = Pubkey::new(&pubkey_vec);
    return admin_pubkey;
}
```

Listing 6: processor/claim_contest_prizes.rs (Lines 127)

```
if winners_header_data_v1.num_prizes_claimed ==
    winners_header_data_v1.num_winner {
    let admin_main_account = next_account_info(account_info_iter)
        ?;

    let admin_pubkey = utils::get_admin_pubkey();
```

Listing 7: processor/create_winners_account_v1.rs (Lines 30)

```
25 if !admin_main_account.is_signer {
26     msg!("ProgramError::MissingRequiredSignature");
27     return Err(ProgramError::MissingRequiredSignature);
28 }
29
30 let admin_pubkey = utils::get_admin_pubkey();
31
32 if admin_pubkey != *admin_main_account.key {
33     msg!("PhantasiaError::AdminDoesNotMatched");
34     return Err(PhantasiaError::AdminDoesNotMatched.into());
35 }
```

Also processor/close_winners_account.rs lines #14-23 and processor/transfer_nft_to_winner.rs lines #19-28.

Risk Level:

Likelihood - 3
Impact - 3

Recommendations:

Implement governance functions to update the admin account address.

Remediation Plan:

ACKNOWLEDGED: The Phantasia Sports team acknowledged this finding.

3.4 (HAL-04) SALARY CAP CONTEST PARAMETER VALIDATION MISSING - MEDIUM

Description:

salary_cap if one of the contest parameters creators specify on initialization. On the front-end, the web application verifies if total roster salary does not exceed the salary_cap for a contest and sends the AddOrUpdateParticipantData instruction. This parameter is not validated again by the program because player IDs in roster_9_players_input arrays do not identify any on-chain objects. In consequence, a malicious participant may create a roster which exceeds the salary_cap for a contest and/or flood the program with invalid participant data thus preventing legitimate users from playing the game and winning awards.

Code Location:

Listing 8: processor/initialize_contest_account.rs (Lines 16)

```rust
pub fn process_initialize_contest_account_v1(
    accounts: &[AccountInfo],
    max_players_in_contest_input: u16,
    first_game_start_time_input: u32,
    contest_league_input: u8,
    salary_cap_input: u16,
    game_type_input: u8,
) -> ProgramResult {
    let account_info_iter = &mut accounts.iter();
```

Listing 9: processor/initialize_contest_account.rs (Lines 58)

```rust
let contest_header_data = ContestInfoHeaderV1 {
    acc_type: AccTypesWithVersion::ContestDataV1 as u8,
    contest_creater_main_pkey: *contest_creater_main_account.key,
    players_joined_contest: 0u16,
    max_players_in_contest: max_players_in_contest_input,
    first_game_start_time: first_game_start_time_input,
    contest_league: contest_league_input,
    salary_cap: salary_cap_input,
    game_type: game_type_input,
};

state::pack_to_contest_info_header_v1(
    contest_header_data,
    &mut contest_data_byte_array[state::CONTEST_HEADER_V1_SI..
        state::CONTEST_HEADER_V1_EI],
)
.unwrap();
```

Listing 10: state.rs (Lines 13,14)

```
12 pub enum ContestLeague {
13     NFL = 1,
14     NBA = 2,
15 }
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendations:

Implement the contest_league_input parameter validation logic in the InitializeContestAccountV1 instruction handler.

Remediation Plan:

ACKNOWLEDGED: The Phantasia Sports team acknowledged this finding.

3.5 (HAL-05) LEAGUE/GAME TYPE CONTEST PARAMETERS VALIDATION MISSING - LOW

Description:

contest_league and game_type are two of the contest parameters creators specify on initialisation. In state.rs, enum ContestLeague defines available leagues. The InitializeContestAccountV1 instruction handler does not ensure the user-supplied contest_league_input is a valid ContestLeague enum variant nor it verifies the game_type_input parameter value. This means users can create contests of arbitrary types and arbitrary games.

Code Location:

Listing 11: processor/initialize_contest_account.rs (Lines 58)

```
52 let contest_header_data = ContestInfoHeaderV1 {
53     acc_type: AccTypesWithVersion::ContestDataV1 as u8,
54     contest_creater_main_pkey: *contest_creater_main_account.key,
55     players_joined_contest: 0u16,
56     max_players_in_contest: max_players_in_contest_input,
57     first_game_start_time: first_game_start_time_input,
58     contest_league: contest_league_input,
59     salary_cap: salary_cap_input,
60     game_type: game_type_input,
61 };
62
63 state::pack_to_contest_info_header_v1(
64     contest_header_data,
65     &mut contest_data_byte_array[state::CONTEST_HEADER_V1_SI..
           state::CONTEST_HEADER_V1_EI],
66 )
67 .unwrap();
```

Listing 12: state.rs (Lines 13,14)

```
pub enum ContestLeague {
    NFL = 1,
    NBA = 2,
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

Implement the contest_league_input parameter validation logic in the InitializeContestAccountV1 instruction handler.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit 28e770a9554267d95348c7c7d6feabde6c3aaffb. The validate_game_and_league_type function was implemented.

3.6 (HAL-06) INTEGER OVERFLOW - LOW

Description:

An overflow happens when an arithmetic operation attempts to create a numeric value that is outside the range that can be represented with a given number of bits. For example, in line #57 in processor/claim\textunderscore contest\textunderscore prizes.rs two u16 values are multiplied without checking whether the result is within the range that can be represented with a given number of bits. If it isn't, in Rust the resulting value is specified to wrap as two's complement, resulting in a value either too low or too high considering the circumstances.

Code Location:

Listing 13: processor/claim_contest_prizes.rs (Lines 57)

```
46 let mut winners_header_data_v1 = state::
       unpack_to_winner_data_header_v1(
47         &winners_storage_data_byte_array
48             [state::WINNER_DATA_HEADER_V1_SI..state::
                   WINNER_DATA_HEADER_V1_EI],
49         true,
50     )?;
51
52     if winners_header_data_v1.acc_type != state::
           AccTypesWithVersion::WinnersDataV1 as u8 {
53         msg!("PhantasiaError::ExpectedAccountTypeMismatched");
54         return Err(PhantasiaError::ExpectedAccountTypeMismatched.
               into());
55     }
56     let winner_data_si: usize = state::WINNER_DATA_HEADER_V1_BYTES
57         + (winner_index_input as usize) * (state::
               WINNER_DATA_V1_BYTES);
58     let winner_data_ei: usize = winner_data_si + state::
           WINNER_DATA_V1_BYTES;
59     let mut current_winner_data: WinnerDataV1 = state::
           unpack_to_winner_data_v1(
60          &winners_storage_data_byte_array[winner_data_si..
                winner_data_ei],
61      )
62      .unwrap();
```

Listing 14: processor/add_or_update_participant _data.rs (Lines 59,60)

```
51 if participant_index_input == 50_000u16 {
52     if now > contest_header_data.first_game_start_time {
53         msg!("PhantasiaError::GameAlreadyStarted");
54         return Err(PhantasiaError::GameAlreadyStarted.into());
55     }
56
57     let participant_data_si = state::
           CONTEST_INFO_HEADER_V1_DATA_BYTES
58         + ((contest_header_data.players_joined_contest as usize)
59             * state::PARTICIPANT_V1_DATA_BYTES);
60     let participant_data_ei = participant_data_si + state::
           PARTICIPANT_V1_DATA_BYTES;
```

Listing 15: processor/add_or_update_participant _data.rs (Lines 82,83)

```
81 let participant_data_si = state::CONTEST_INFO_HEADER_V1_DATA_BYTES
82         + (participant_index_input as usize) * state::
               PARTICIPANT_V1_DATA_BYTES;
83     let participant_data_ei = participant_data_si + state::
           PARTICIPANT_V1_DATA_BYTES;
84     let mut participant_data = state::unpack_to_participant_v1(
85         &contest_data_byte_array[participant_data_si..
               participant_data_ei],
86     )
87     .unwrap();
88     if participant_data.participant_main_pkey != *
           participant_main_account.key {
89         msg!("PhantasiaError::WrongParticipantIndex");
90         return Err(PhantasiaError::WrongParticipantIndex.into());
91     }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

Consider using the checked_add and checked_mul methods instead of addition and multiplication operators respectively, to handle overflows gracefully.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit 28e770a9554267d95348c7c7d6feabde6c3aaffb.

3.7 (HAL-07) INDEX OUT OF BOUNDS - LOW

Description:

To create a Solana account, a client generates a keypair and registers its public key using the CreateAccount system instruction with a **fixed** storage size in bytes preallocated.

The following instruction handlers:
- InitializeJoiningFeeCollectionRecordAccount
- InitializeCircularlySortedTimestampStorageAccount
- InitializeStakeUsersStorageAccount

expect some user-supplied accounts storage sizes to be at least two bytes but do not verify it which will lead to program panic if the size is lower than two bytes.

Code Location:

Listing 16: processor.rs (Lines 1653,1654)

```rust
let league_entry_fee_collection_storage_account =
    next_account_info(account_info_iter)?;

let rent = Rent::get()?;

if !rent.is_exempt(
    league_entry_fee_collection_storage_account.lamports(),
    league_entry_fee_collection_storage_account.data_len(),
) {
    return Err(PhantasiaError::NotRentExempt.into());
}

let mut league_entry_fee_collection_byte_array_data =
    league_entry_fee_collection_storage_account
        .data
        .borrow_mut();

if league_entry_fee_collection_byte_array_data[0] != 0 {
    return Err(PhantasiaError::
        LeagueEntryFeeCollectionAccountAlreadyInitialized.into());
}

league_entry_fee_collection_byte_array_data[0] = 1u8;
league_entry_fee_collection_byte_array_data[1] = 6u8;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendations:

Validate the user-supplied account's storage size to match the expected one.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit 28e770a9554267d95348c7c7d6feabde6c3aaffb. This code was removed from the program.

3.8 (HAL-08) POSSIBLE MISUSE OF HELPER METHODS - INFORMATIONAL

Description:

The intention and use of helper methods in Rust, like unwrap, is very useful for testing environments because a value is forcibly demanded to get an error (aka panic!) if the Option the methods is called on doesn't have Some value or Result. Nevertheless, leaving unwrap functions in production environments is a bad practice because not only will this cause the program to crash out, or panic!. In addition, no helpful messages are shown to help the user solve, or understand the reason of the error.

Code Location:

Note: some usages of unwrap are justified and were excluded from the listing below.

Listing 17

```
auditor@halborn:~/src/phantasia/$ git grep -n 'unwrap()'
nstruction.rs:108:                    roster_9_players_input:
    u8_array_to_u32_array(&input[1..37].try_into().unwrap()),
processor/add_or_update_participant_data.rs:34:     let mut
    contest_data_byte_array = contest_account.data.try_borrow_mut()
    .unwrap();
processor/add_or_update_participant_data.rs:41:     .unwrap();
processor/add_or_update_participant_data.rs:68:         .unwrap();
processor/add_or_update_participant_data.rs:79:         .unwrap();
processor/add_or_update_participant_data.rs:87:         .unwrap();
processor/add_or_update_participant_data.rs:98:         .unwrap();
processor/claim_contest_prizes.rs:39:
    winners_storage_account.data.try_borrow_mut().unwrap();
processor/claim_contest_prizes.rs:62:     .unwrap();
processor/claim_contest_prizes.rs:108:        .unwrap();
processor/claim_contest_prizes.rs:121:    .unwrap();
processor/claim_contest_prizes.rs:139:        .unwrap();
processor/claim_contest_prizes.rs:146:        .unwrap();
```

```
processor/close_winners_account.rs:34:    .unwrap();
processor/create_winners_account_v1.rs:50:    let mut contest_data_byte_array = contest_account.data.try_borrow_mut().unwrap();
processor/create_winners_account_v1.rs:56:    .unwrap();
processor/create_winners_account_v1.rs:82: winners_storage_account.data.try_borrow_mut().unwrap();
processor/create_winners_account_v1.rs:119:        .unwrap();
processor/create_winners_account_v1.rs:131:        .unwrap();
processor/create_winners_account_v1.rs:139:            .unwrap() as u64;
processor/create_winners_account_v1.rs:142:            .unwrap();
processor/create_winners_account_v1.rs:185:            .unwrap();
processor/create_winners_account_v1.rs:213:        .unwrap();
processor/create_winners_account_v1.rs:221:    .unwrap();
processor/initialize_contest_account_v1.rs:38:            .unwrap();
processor/initialize_contest_account_v1.rs:45:    let mut contest_data_byte_array = contest_account.data.try_borrow_mut().unwrap();
processor/initialize_contest_account_v1.rs:67:    .unwrap();
processor/transfer_nft_to_winner.rs:94:    .unwrap();
processor/update_or_create_fan_card_v1.rs:103:    let mut fan_card_data_byte_array = fan_card_storage_account.data.try_borrow_mut().unwrap();
processor/update_or_create_fan_card_v1.rs:136:        .copy_from_slice(&user_fan_card.try_to_vec().unwrap());
utils.rs:15:    let pubkey_vec = bs58::decode(admin_pubkey_str).into_vec().unwrap();
utils.rs:25:    return bytes.try_into().unwrap();
utils.rs:31:        bytes.push(u32::from_le_bytes(inp[ind..ind + 4].try_into().unwrap()));
utils.rs:33:    return bytes.try_into().unwrap();
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

It is recommended not use the unwrap function in production environment because this use provokes panic! and may crash the contract without verbose error messages. Crashing the system will result in a loss of availability, and in some cases, even private information stored in the state. Some alternatives are possible, such as propagating the error with ? instead of unwrap or using the error-chain crate for errors.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit 067f426ea3c75a733f0765791f010f333da00d5b.

3.9 (HAL-09) MISSING CONTEST CREATOR VALIDATION - INFORMATIONAL

Description:

On account initialization, the initialize_contest_account_v1 function saves the contest creator public key as contest_creater_main_pkey . Two of the accounts the CreateWinnersAccountV1 instruction handler requires the caller to provide are contest_account and contest_creater_main_account. The handler does not validate if the user-supplied contest_creater_main_account parameter value matches the contest_creater_main_pkey property of contest_account. This means users can provide arbitrary addresses as contest_creater_main_accounts.

Code Location:

Listing 18: processor/initialize_contest_account_v1.rs (Lines 54)

```
52 let contest_header_data = ContestInfoHeaderV1 {
53     acc_type: AccTypesWithVersion::ContestDataV1 as u8,
54     contest_creater_main_pkey: *contest_creater_main_account.key,
55     players_joined_contest: 0u16,
56     max_players_in_contest: max_players_in_contest_input,
57     first_game_start_time: first_game_start_time_input,
58     contest_league: contest_league_input,
59     salary_cap: salary_cap_input,
60     game_type: game_type_input,
61 };
62
63 state::pack_to_contest_info_header_v1(
64     contest_header_data,
65     &mut contest_data_byte_array[state::CONTEST_HEADER_V1_SI..
           state::CONTEST_HEADER_V1_EI],
66 )
67 .unwrap();
```

Listing 19: create_winners_account_v1.rs (Lines 41)

```
let contest_account = next_account_info(account_info_iter)?;

    let winners_storage_account = next_account_info(
        account_info_iter)?;

    let contest_creater_main_account = next_account_info(
        account_info_iter)?;

    let rent = Rent::get()?;

    if contest_account.owner != program_id {
        msg!("PhantasiaError::WrongAccountPassed");
        return Err(PhantasiaError::WrongAccountPassed.into());
    }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Implement validation of the contest_creater_main_account address in the CreateWinnersAccountV1 instruction handler.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit 28e770a9554267d95348c7c7d6feabde6c3aaffb. The instruction handler validates the contest creator now.

3.10 (HAL-10) CREATING FAN CARDS FROM FROZEN NFTS ALLOWED - INFORMATIONAL

Description:

SPL Token accounts can be frozen. When they are, no SPL Token instructions can be executed on such accounts--they can't be transferred and their properties cannot be modified. The UpdateOrCreateFanCardV1 instruction handler does not verify if any of the NFT accounts provided is frozen before updating or creating a fan card.

Code Location:

Listing 20: processor/update_or_create_fan_card_.rs (Lines 170,175,180)

```
167 let nft_ata_account_info = TokenAccount::unpack(&nft_ata.data.
        borrow())?;
168 let nft_mint_info = Mint::unpack(&nft_mint.data.borrow_mut())?;
169
170 if nft_ata_account_info.owner != *user_wallet_pubkey {
171     msg!("PhantasiaError::NftAtaAccountOwnerMismatched");
172     return Err(PhantasiaError::NftAtaAccountOwnerMismatched.into()
            );
173 }
174
175 if nft_ata_account_info.mint != *nft_mint.key {
176     msg!("PhantasiaError::PassedATAmintAndNFTmintMismatched");
177     return Err(PhantasiaError::PassedATAmintAndNFTmintMismatched.
            into());
178 }
179
180 if nft_ata_account_info.amount != 1 {
181     msg!("PhantasiaError::NftAtaAccountDoesNotHaveNFT");
182     return Err(PhantasiaError::NftAtaAccountDoesNotHaveNFT.into())
            ;
183 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Implement validation of the NFT accounts' state parameters in the UpdateOrCreateFanCardV1 instruction handler.

Remediation Plan:

SOLVED: The Phantasia Sports team fixed the issue in commit 28e770a9554267d95348c7c7d6feabde6c3aaffb. The NFT validation includes Token account state check now.

AUTOMATED TESTING

4.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and vulnerabilities. Among the tools used was cargo audit, a security scanner for vulnerabilities reported to the RustSec 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 auditors 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.

ID	**package**	**Short Description**
RUSTSEC-2020-0159	chrono	Potential segfault in ‘localtime_r‘ invocations
RUSTSEC-2021-0079	hyper	Integer overflow in ‘hyper‘’s parsing of the ‘Transfer-Encoding‘ header leads to data loss
RUSTSEC-2021-0078	hyper	Lenient ‘hyper‘ header parsing of ‘Content-Length‘ could allow request smuggling
RUSTSEC-2021-0119	nix	Out-of-bounds write in nix::unistd::getgrouplist
RUSTSEC-2020-0071	time	Potential segfault in the time crate

THANK YOU FOR CHOOSING

// HALBORN