// Security Assessment

03.31.2025 - 04.10.2025

MO - Solana Program

MO

HALBORN

M0 - Solana Program - M0

Prepared by: HALBORN

Last Updated 04/17/2025

Date of Engagement: March 31st, 2025 - April 10th, 2025

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
11	0	0	1	5	5

TABLE OF CONTENTS

8. Automated Testing

1. Introduction

M^0 Foundation team engaged Halborn to conduct a security assessment on their **Earn, Ext_Earn and Portal Solana programs** beginning on March 1st, 2025, and ending on April 11th, 2025. The security assessment was scoped to the Solana Programs provided in solana-m GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

The Solana M platform is a system for managing and distributing yield to token holders through multiple coordinated programs. The purpose of the system is to allow bridging M token to Solana and maintaining the yield earning features found on EVM chains. The protocol's core consists of the following programs:

- Earn : Handles yield distribution logic and earner management for the M token. The features include yield distribution cycles, earner registration via Merkle proofs and claim processing.
- ExtEarn : Handles wrapping/unwrapping M to wM as well as yield distribution and earner manager for the wM token.
- Portal : The Portal is a fork of the Wormhole Native Token Transfer (NTT) program with a few modifications including custom payload, utilizing token multisig mint authority and added CPI call to the earn program to store custom data sent in the payload.

2. Assessment Summary

Halborn was provided 9 days for the engagement and assigned one full-time security engineer to review the security of the Solana Programs in scope. The engineer is a blockchain and smart contract security expert with advanced smart contract hacking skills, and deep knowledge of multiple blockchain protocols.

The purpose of the assessment is to:

- Identify potential security issues within the Solana Programs.
- Ensure that smart contract functionality operates as intended.

In summary, Halborn identified some security concerns and improvements, that have been partially addressed by the M^0 Foundation team . The main ones were the following:

- Enforce the number of extension mint's decimals to be the same as the M0 mint's decimals.
- Correctly validate the M0 mint during initialization.
- Implement access control during programs initialization.
- Implement access control when removing orphaned earner accounts.
- Validate the enabled mint token extensions.
- Do not allow a removed earn manager to set earner's recipient addresses.

3. Test Approach And 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 program assessment. While manual testing is recommended to uncover flaws in business logic, processes, 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 assessment:

- Research into the architecture, purpose, and use of the platform.

- Manual program 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.

- Scanning dependencies for known vulnerabilities (`cargo audit`).

- Local runtime testing (`anchor test`)

4. 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.

4.1 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) Specific (AO:S)	1 0.2

EXPLOITABILITY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33
Attack Complexity (AX)	Low (AX:L) Medium (AX:M) High (AX:H)	1 0.67 0.33

Exploitability E is calculated using the following formula:

$$E = \prod m_e$$

4.2 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) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Integrity (I)	None (I:N) Low (I:L) Medium (I:M) High (I:H) Critical (I:C)	0 0.25 0.5 0.75 1
Availability (A)	None (A:N) Low (A:L) Medium (A:M) High (A:H) Critical (A:C)	0 0.25 0.5 0.75 1
Deposit (D)	None (D:N) Low (D:L) Medium (D:M) High (D:H) Critical (D:C)	0 0.25 0.5 0.75 1
Yield (Y)	None (Y:N) Low (Y:L) Medium (Y:M) High (Y:H) Critical (Y:C)	0 0.25 0.5 0.75 1

Impact I is calculated using the following formula:

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

4.3 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.

METRICS:

SEVERITY COEFFICIENT (C)	COEFFICIENT VALUE	NUMERICAL VALUE
Reversibility (r)	None (R:N) Partial (R:P) Full (R:F)	1 0.5 0.25
Scope (s)	Changed (S:C) Unchanged (S:U)	1.25 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

SEVERITY	SCORE VALUE RANGE
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: solana-m

(b) Assessed Commit ID: 2d8e2af

(c) Items in scope:

- ./programs/portal/Cargo.toml
- ./programs/portal/Xargo.toml
- ./programs/portal/src/instructions/luts.rs
- ./programs/portal/src/instructions/mark_outbox_item_as_released.rs
- ./programs/portal/src/instructions/redeem.rs
- ./programs/portal/src/instructions/initialize.rs
- ./programs/portal/src/instructions/mod.rs
- ./programs/portal/src/instructions/release_inbound.rs
- ./programs/portal/src/instructions/transfer.rs
- ./programs/portal/src/instructions/admin.rs
- ./programs/portal/src/pending_token_authority.rs
- ./programs/portal/src/clock.rs
- ./programs/portal/src/error.rs
- ./programs/portal/src/config.rs
- ./programs/portal/src/lib.rs
- ./programs/portal/src/payloads/mod.rs
- ./programs/portal/src/payloads/token_transfer.rs
- ./programs/portal/src/bitmap.rs
- ./programs/portal/src/transceivers/wormhole/instructions/broadcast_id.rs
- ./programs/portal/src/transceivers/wormhole/instructions/broadcast_peer.rs
- ./programs/portal/src/transceivers/wormhole/instructions/mod.rs
- ./programs/portal/src/transceivers/wormhole/instructions/release_outbound.rs
- ./programs/portal/src/transceivers/wormhole/instructions/receive_message.rs
- ./programs/portal/src/transceivers/wormhole/instructions/admin.rs
- ./programs/portal/src/transceivers/wormhole/accounts.rs
- ./programs/portal/src/transceivers/wormhole/mod.rs
- ./programs/portal/src/transceivers/mod.rs
- ./programs/portal/src/transceivers/accounts/mod.rs
- ./programs/portal/src/transceivers/accounts/peer.rs
- ./programs/portal/src/queue/outbox.rs
- ./programs/portal/src/queue/mod.rs
- ./programs/portal/src/queue/rate_limit.rs
- ./programs/portal/src/queue/inbox.rs
- ./programs/portal/src/registered_transceiver.rs

- ./programs/portal/src/spl_multisig.rs
- ./programs/portal/src/messages.rs
- ./programs/portal/src/peer.rs
- ./programs/earn/Cargo.toml
- ./programs/earn/Xargo.toml
- ./programs/earn/src/instructions/portal/mod.rs
- ./programs/earn/src/instructions/portal/propagate_index.rs
- ./programs/earn/src/instructions/earn_authority/claim_for.rs
- ./programs/earn/src/instructions/earn_authority/complete_claims.rs
- ./programs/earn/src/instructions/earn_authority/mod.rs
- ./programs/earn/src/instructions/admin/initialize.rs
- ./programs/earn/src/instructions/admin/mod.rs
- ./programs/earn/src/instructions/admin/set_earn_authority.rs
- ./programs/earn/src/instructions/admin/set_claim_cooldown.rs
- ./programs/earn/src/instructions/mod.rs
- ./programs/earn/src/instructions/open/remove_registrar_earner.rs
- ./programs/earn/src/instructions/open/add_registrar_earner.rs
- ./programs/earn/src/instructions/open/mod.rs
- ./programs/earn/src/constants.rs
- ./programs/earn/src/lib.rs
- ./programs/earn/src/utils/token.rs
- ./programs/earn/src/utils/mod.rs
- ./programs/earn/src/utils/merkle_proof.rs
- ./programs/earn/src/state/global.rs
- ./programs/earn/src/state/earner.rs
- ./programs/earn/src/state/mod.rs
- ./programs/earn/src/errors.rs
- ./programs/ext_earn/Cargo.toml
- ./programs/ext_earn/Xargo.toml
- ./programs/ext_earn/src/instructions/earn_authority/claim_for.rs
- ./programs/ext_earn/src/instructions/earn_authority/sync.rs
- ./programs/ext_earn/src/instructions/earn_authority/mod.rs
- ./programs/ext_earn/src/instructions/admin/add_earn_manager.rs
- ./programs/ext_earn/src/instructions/admin/initialize.rs
- ./programs/ext_earn/src/instructions/admin/mod.rs
- ./programs/ext_earn/src/instructions/admin/remove_earn_manager.rs
- ./programs/ext_earn/src/instructions/admin/set_earn_authority.rs
- ./programs/ext_earn/src/instructions/earner/set_recipient.rs
- ./programs/ext_earn/src/instructions/earner/mod.rs
- ./programs/ext_earn/src/instructions/mod.rs
- ./programs/ext_earn/src/instructions/open/unwrap.rs
- ./programs/ext_earn/src/instructions/open/remove_orphaned_earner.rs
- ./programs/ext_earn/src/instructions/open/mod.rs
- ./programs/ext_earn/src/instructions/open/wrap.rs

- ./programs/ext_earn/src/instructions/earn_manager/configure.rs
- ./programs/ext_earn/src/instructions/earn_manager/mod.rs
- ./programs/ext_earn/src/instructions/earn_manager/add_earner.rs
- ./programs/ext_earn/src/instructions/earn_manager/transfer_earner.rs
- ./programs/ext_earn/src/instructions/earn_manager/remove_earner.rs
- ./programs/ext_earn/src/constants.rs
- ./programs/ext_earn/src/lib.rs
- ./programs/ext_earn/src/utils/token.rs
- ./programs/ext_earn/src/utils/mod.rs
- ./programs/ext_earn/src/state/global.rs
- ./programs/ext_earn/src/state/earn_manager.rs
- ./programs/ext_earn/src/state/earner.rs
- ./programs/ext_earn/src/state/mod.rs
- ./programs/ext_earn/src/errors.rs

Out-of-Scope: Third party dependencies and economic attacks.

REMEDIATION COMMIT ID:

- https://github.com/m0-foundation/solana-m/pull/52
- https://github.com/m0-foundation/solana-m/pull/65
- https://github.com/m0-foundation/solana-m/pull/56

Out-of-Scope: New features/implementations after the remediation commit IDs.

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	5	5

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
EXTENSION MINT DECIMALS ARE NOT ENFORCED	MEDIUM	SOLVED - 04/08/2025

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
TOKEN 2022 EXTENSIONS ARE NOT VALIDATED	LOW	RISK ACCEPTED - 04/15/2025
ANYONE CAN CALL THE REMOVE_ORPHANED_EARNER INSTRUCTION AND REDEEM ACCOUNT RENT FEE	LOW	RISK ACCEPTED - 04/15/2025
RISK OF FRONT-RUNNING DURING PROGRAM INITIALIZATION	LOW	RISK ACCEPTED - 04/15/2025
THE MO MINT IS NOT VALIDATED DURING INITIALIZATION	LOW	SOLVED - 04/15/2025
REMOVED EARN MANAGER CAN SET EARNER'S RECIPIENT	LOW	RISK ACCEPTED - 04/15/2025
EARNER ACCOUNTS CAN BE TRANSFERRED ONLY BY AN ACTIVE EARN MANAGER	INFORMATIONAL	ACKNOWLEDGED - 04/15/2025
THE PROGRAM RELIES ON OFF-CHAIN LOGIC	INFORMATIONAL	ACKNOWLEDGED - 04/15/2025
UNUSED INSTRUCTION WITH VULNERABILITY SURFACE	INFORMATIONAL	SOLVED - 04/09/2025
TWO-STEP EARN MANAGER AUTHORITY TRANSFER NOT IMPLEMENTED	INFORMATIONAL	ACKNOWLEDGED - 04/15/2025
TWO-STEP EARN AUTHORITY TRANSFER NOT IMPLEMENTED	INFORMATIONAL	ACKNOWLEDGED - 04/15/2025

7. FINDINGS & TECH DETAILS

7.1 EXTENSION MINT DECIMALS ARE NOT ENFORCED

// MEDIUM

Description

The `initialize` instruction of the `ext_earn` program is responsible for setting up the program configuration, including specifying a new `ext_mint` extension mint. However, the instruction does not enforce that the `decimals` value of the `ext_mint` matches the `decimals` of the main M0 token mint.

Risk Factors:

- If the `ext_mint` is configured with a different number of decimals than the M0 mint, the system will be unable to perform accurate 1:1 mappings between the two tokens.
- This discrepancy can lead to incorrect collateralization calculations, which may result in over- or under-collateralization of assets.
- Additionally, yield distributions and other token-based computations could become inaccurate, potentially leading to financial loss, unexpected behavior, or exploitable edge cases in the protocol.

programs/ext_earn/src/instructions/admin/initialize.rs

```
#[derive(Accounts)]
pub struct Initialize<'info> {
    #[account(mut)]
    pub admin: Signer<'info>,

    #[account(
        init,
        payer = admin,
        space = ANCHOR_DISCRIMINATOR_SIZE + ExtGlobal::INIT_SPACE,
        seeds = [EXT_GLOBAL_SEED],
        bump
    )]
    pub global_account: Account<'info, ExtGlobal>,

    #[account(
        token::token_program = token_2022
    )]
    pub m_mint: InterfaceAccount<'info, Mint>,

    #[account(
        token::token_program = token_2022
    )]
    pub ext_mint: InterfaceAccount<'info, Mint>,

    #[account(
        seeds = [EARN_GLOBAL_SEED],
        seeds::program = EARN_PROGRAM,
        bump = m_earn_global_account.bump,
    )]
    pub m_earn_global_account: Account<'info, EarnGlobal>,

    pub token_2022: Program<'info, Token2022>,

    pub system_program: Program<'info, System>,
}`
```

The severity of this finding is currently rated as **Medium** because the `initialize` instruction lacks access control, allowing any user to initialize the program—even with incorrect mint decimals.
However, the risk is somewhat mitigated by the fact that the `ext_earn` program is a **first-party extension** of the M0 token. This close integration makes it **highly unlikely** that an unauthorized or incorrect initialization would go unnoticed by the development or deployment team.

Proof of Concept

The test case below passess successfully and initializes the ext_mint mint account with 12 decimals.

```
test("HALBORN initialize incorrect decimals - success", async () => {
        // Setup the instruction call
        const wrongMint = new Keypair();
        // the createMint function was adapted to set custom mint decimals
        await createMint(wrongMint, nonAdmin.publicKey, true, 12);

        const { globalAccount } = prepExtInitialize(admin);

        // pass the wrong mint to the initialize instruction
        accounts.extMint = wrongMint.publicKey;
        // Create and send the transaction
        await extEarn.methods
          .initialize(earnAuthority.publicKey)
          .accounts({ ...accounts })
          .signers([admin])
          .rpc();

        // Calculate the expected bumps
        const [, globalBump] = PublicKey.findProgramAddressSync(
          [Buffer.from("global")],
          EXT_EARN_PROGRAM_ID
        );
        const [, mVaultBump] = PublicKey.findProgramAddressSync(
          [Buffer.from("m_vault")],
          EXT_EARN_PROGRAM_ID
        );
        const [, extMintAuthorityBump] = PublicKey.findProgramAddressSync(
          [Buffer.from("mint_authority")],
          EXT_EARN_PROGRAM_ID
        );

        await expectExtGlobalState(globalAccount, {
          admin: admin.publicKey,
          mMint: mMint.publicKey,
          extMint: wrongMint.publicKey,
          earnAuthority: earnAuthority.publicKey,
          index: initialIndex,
          timestamp: new BN(svm.getClock().unixTimestamp.toString()),
          bump: globalBump,
          mVaultBump,
          extMintAuthorityBump,
        });
      });
```

BVSS

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

Recommendation

To address this issue, it is recommended to ensure that the `decimals` of the `ext_mint` match those of the M0 mint, preserving consistency across token operations.

Remediation Comment

SOLVED: The **M0 team** solved the finding by ensuring `ext_mint` token decimals matches `m_mint` decimals.

Remediation Hash

https://github.com/m0-foundation/solana-m/pull/52

7.2 TOKEN 2022 EXTENSIONS ARE NOT VALIDATED

// LOW

Description

The `earn::initialize`, `ext_earn::initialize`, and `portal::initialize_multisig` instructions allow callers to initialize and configure their respective programs. As part of this process, they also set the mints for the core M0 token or associated extension tokens. However, these instructions do **not** verify whether the provided mint has Token-2022 extensions enabled—potentially introducing a range of risks depending on the active extensions.
For example:

- **TransferFee Extension**: This extension deducts a fee from each token transfer, meaning the actual transferred amount is less than expected. In the context of the `transfer_burn` instruction, this could cause a larger amount of tokens to be burned from the custody account than was truly transferred—leading to inconsistencies, denial of service, or even loss of funds.
- **Non-Transferable Extension**: If this extension is active, it would prevent tokens from being moved in or out of the custody account, resulting in a denial of service.

Although it is anticipated that these programs will be initialized by a trusted authority, the current implementations of `earn` and `ext_earn` lack access control, allowing **any** user to initialize them with potentially malicious mints. Even after limiting initialization to trusted authorities, there remains a risk that a mint with dangerous extensions could still be configured, compromising the integrity of the protocol.

BVSS

AO:A/AC:L/AX:L/R:P/S:U/C:H/A:M/I:N/D:N/Y:N (4.4)

Recommendation

To address this issue, it is recommended to validate the active Token-2022 extensions on the provided mint and explicitly reject any extensions that could pose risks or interfere with the expected behavior of the protocol—such as the `TransferFee` or `NonTransferable` extensions.

Remediation Comment

RISK ACCEPTED: The **M0 team** accepted the risk, noting that initialization will occur immediately after deployment under their control. They are not concerned about the tokens having extensions affecting functionality but will keep this in mind for future updates.

7.3 ANYONE CAN CALL THE REMOVE_ORPHANED_EARNER INSTRUCTION AND REDEEM ACCOUNT RENT FEE

// LOW

Description

The `remove_orphaned_earner` instruction allows for the closure of earner accounts associated with a removed earn manager. However, it lacks any access control, meaning that anyone can invoke it. This poses potential risks—for instance, if an earn manager is intended to be deactivated only temporarily, a malicious actor could exploit this period to close associated earner accounts and claim their rent-exempt balances, resulting in unintended loss of state and disruption to the protocol.

programs/ext_earn/src/instructions/open/remove_orphaned_earner.rs

```
14 #[derive(Accounts)]
15 pub struct RemoveOrphanedEarner<'info> {
16     #[account(mut)]
17     pub signer: Signer<'info>,
18
19     #[account(
20         seeds = [EXT_GLOBAL_SEED],
21         bump = global_account.bump,
22     )]
23     pub global_account: Account<'info, ExtGlobal>,
24
25     #[account(
26         mut,
27         close = signer,
28         seeds = [EARNER_SEED, earner_account.user_token_account.as_ref()],
29         bump = earner_account.bump,
30     )]
31     pub earner_account: Account<'info, Earner>,
32
33     #[account(
34         constraint = !earn_manager_account.is_active @ ExtError::Active,
35         seeds = [EARN_MANAGER_SEED, earner_account.earn_manager.as_ref()],
36         bump = earn_manager_account.bump,
37     )]
38     pub earn_manager_account: Account<'info, EarnManager>,
39
40     pub system_program: Program<'info, System>,
41 }
42
43 pub fn handler(_ctx: Context<RemoveOrphanedEarner>) -> Result<()> {
44     Ok(())
45 }
```

BVSS

AO:A/AC:L/AX:L/R:P/S:U/C:N/A:M/I:M/D:L/Y:N (3.4)

Recommendation

To address this issue, it is recommended to restrict the `remove_orphaned_earner` instruction access control to a known defined authority.

Remediation Comment

RISK ACCEPTED: The **MO team** accepts the risk and confirms this behavior is by design. It serves as an incentive to clean up orphaned earners in cases where an Earn Manager is removed by the admin, rather than the manager voluntarily exiting the system via `remove_earner` and claiming the refund.

7.4 RISK OF FRONT-RUNNING DURING PROGRAM INITIALIZATION

// LOW

Description

The `earn::initialize` and `ext_earn::initialize` instructions are designed to initialize the respective programs and configure their settings. However, the instructions currently lack proper access control—they do not restrict or validate the signer. As a result, any user can invoke it, creating a risk that a malicious actor could front-run the intended administrator and initialize the program with unauthorized settings.

programs/earn/src/instructions/admin/initialize.rs

```
#[derive(Accounts)]
pub struct Initialize<'info> {
    #[account(mut)]
    pub admin: Signer<'info>,

    #[account(
        init,
        payer = admin,
        space = ANCHOR_DISCRIMINATOR_SIZE + Global::INIT_SPACE,
        seeds = [GLOBAL_SEED],
        bump
    )]
    pub global_account: Account<'info, Global>,

    pub system_program: Program<'info, System>,
}
```

programs/ext_earn/src/instructions/admin/initialize.rs

```
#[derive(Accounts)]
pub struct Initialize<'info> {
    #[account(mut)]
    pub admin: Signer<'info>,

    #[account(
        init,
        payer = admin,
        space = ANCHOR_DISCRIMINATOR_SIZE + ExtGlobal::INIT_SPACE,
        seeds = [EXT_GLOBAL_SEED],
        bump
    )]
    pub global_account: Account<'info, ExtGlobal>,

    #[account(
        token::token_program = token_2022
    )]
    pub m_mint: InterfaceAccount<'info, Mint>
// ...
}
```

BVSS

AO:A/AC:L/AX:L/R:P/S:U/C:N/A:M/I:L/D:L/Y:N (3.1)

Recommendation

To address this issue, it is recommended to implement a proper access control and validate that the signer is a known authorized address.

Remediation Comment

RISK ACCEPTED: The **MO team** accepted the risk, noting that in the event of initialization being frontrun, the programs will be re-deployed.

7.5 THE MO MINT IS NOT VALIDATED DURING INITIALIZATION

// LOW

Description

The `earn::initialize` instruction is used to initialize and configure the earn program. One of the configuration parameters it sets is the `mint` field, which is expected to be the mint address of the MO token. However, this address is only passed as an instruction argument and is neither validated for correctness nor checked to ensure it refers to a valid mint account.

In addition, the `ext_earn::initialize` instruction is used to initialize and configure the ext_earn program. During the initialization, it is expected that the earn program will be already initialized. However, the instruction does not verify, that the provided `m_mint` account actually corresponds to the MO token mint set during the initialization of the earn program.

As a result, unintentionally providing incorrect mint addresses could lead to inconsistencies and denial of service. Furthermore, since there is no instruction to update the configuration after initialization, this mistake cannot be easily corrected later.

programs/earn/src/instructions/admin/initialize.rs

```
31 pub fn handler(
32     ctx: Context<Initialize>,
33     mint: Pubkey,
34     earn_authority: Pubkey,
35     initial_index: u64,
36     claim_cooldown: u64,
37 ) -> Result<()> {
38     // Check that the initial index is at least 1 (with 12 decimals)
39     if initial_index < 1_000_000_000_000 {
40         return err!(EarnError::InvalidParam);
41     }
42 
43     // Check that the claim cooldown is not longer than 1 week
44     if claim_cooldown > 604800 {
45         return err!(EarnError::InvalidParam);
46     }
47 
48     // Portal authority that will propagate index and roots
49     let portal_authority = Pubkey::find_program_address(&[TOKEN_AUTHORITY_SEED], &PORTAL_PROGRAM).0;
50 
51     ctx.accounts.global_account.set_inner(Global {
52         admin: ctx.accounts.admin.key(),
53         earn_authority,
54         portal_authority,
55         mint,
56         index: initial_index,
57         timestamp: 0, // Set this to 0 initially so we can call propagate immediately
58         claim_cooldown,
59         max_supply: 0,
60         max_yield: 0,
61         distributed: 0,
62         claim_complete: true,
63         earner_merkle_root: [0; 32],
64         bump: ctx.bumps.global_account,
65     });
66 
67     Ok(())
68 }
```

programs/ext_earn/src/instructions/admin/initialize.rs

```rust
#[derive(Accounts)]
pub struct Initialize<'info> {
    #[account(mut)]
    pub admin: Signer<'info>,

    #[account(
        init,
        payer = admin,
        space = ANCHOR_DISCRIMINATOR_SIZE + ExtGlobal::INIT_SPACE,
        seeds = [EXT_GLOBAL_SEED],
        bump
    )]
    pub global_account: Account<'info, ExtGlobal>,

    #[account(
        token::token_program = token_2022
    )]
    pub m_mint: InterfaceAccount<'info, Mint>,

    #[account(
        token::token_program = token_2022
    )]
    pub ext_mint: InterfaceAccount<'info, Mint>,

    #[account(
        seeds = [EARN_GLOBAL_SEED],
        seeds::program = EARN_PROGRAM,
        bump = m_earn_global_account.bump,
    )]
    pub m_earn_global_account: Account<'info, EarnGlobal>,

    pub token_2022: Program<'info, Token2022>,

    pub system_program: Program<'info, System>,
}
```

BVSS

AO:A/AC:L/AX:L/R:P/S:U/C:N/A:M/I:L/D:N/Y:N (2.8)

Recommendation

To address this issue, it is recommended to validate that the provided address is a valid Mint account. Alternatively, the instruction could enforce the use of a predefined, hardcoded address corresponding to the official M0 token mint. Also, ensure that the `m_mint` account provided during the ext_earn program initialization actually corresponds to the mint of the earn program configuration.

Remediation Comment

SOLVED: The **M0 team** solved the finding by validating the mint account in the `earn::initialize` instruction. Additionally, the `m_mint` provided in `ext_earn::initialize` is now enforced to match the mint configured in the Earn program.

Remediation Hash

https://github.com/m0-foundation/solana-m/pull/65

7.6 REMOVED EARN MANAGER CAN SET EARNER'S RECIPIENT

// LOW

Description

The `set_recipient` instruction allows either the associated user or an earn manager to set the recipient account for an earner. However, the instruction does not verify whether the earn manager account is still active or has been previously removed. As a result, even removed earn managers retain the ability to set recipient accounts.
This lack of validation introduces a risk where a compromised or malicious earn manager could set recipient accounts for arbitrary earners, potentially resulting in misdirected payouts or loss of funds.

<file name>.rs

```
15 #[derive(Accounts)]
16 pub struct SetRecipient<'info> {
17     #[account(
18         constraint =
19             signer.key() == earner_account.user ||
20             signer.key() == earner_account.earn_manager
21             @ ExtError::NotAuthorized,
22     )]
23     pub signer: Signer<'info>,
24
25     #[account(
26         seeds = [EXT_GLOBAL_SEED],
27         bump = global_account.bump,
28     )]
29     pub global_account: Account<'info, ExtGlobal>,
30
31     #[account(
32         mut,
33         seeds = [EARNER_SEED, &earner_account.user_token_account.as_ref()],
34         bump = earner_account.bump,
35     )]
36     pub earner_account: Account<'info, Earner>,
37
38     #[account(
39         token::mint = global_account.ext_mint,
40         constraint = has_immutable_owner(&recipient_token_account) @ ExtError::MutableOwner,
41     )]
42     pub recipient_token_account: Option<InterfaceAccount<'info, TokenAccount>>,
43 }
```

BVSS

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

Recommendation

To address this issue, it is recommended to allow only active managers to set an earner's recipient.

Remediation Comment

RISK ACCEPTED: The **M0 team** accepts the risk, noting that if the earn manager is inactive, the associated earner is treated as orphaned and can be removed from the system.

7.7 EARNER ACCOUNTS CAN BE TRANSFERRED ONLY BY AN ACTIVE EARN MANAGER

// INFORMATIONAL

Description

The `transfer_earner` instruction allows an active earn manager to transfer a user's earner account to another active earn manager.
However, if an earn manager has been removed and is no longer active, there is currently no way to transfer the associated earner accounts to another manager. The only available option is to re-activate the removed manager, which could pose a risk by restoring privileges to a potentially compromised or malicious manager.

<file name>.rs

```
13 #[derive(Accounts)]
14 #[instruction(to_earn_manager: Pubkey)]
15 pub struct TransferEarner<'info> {
16     pub signer: Signer<'info>,
17
18     #[account(
19         mut,
20         constraint = earner_account.earn_manager == signer.key() @ ExtError::NotAuthorized,
21         seeds = [EARNER_SEED, earner_account.user_token_account.as_ref()],
22         bump = earner_account.bump,
23     )]
24     pub earner_account: Account<'info, Earner>,
25
26     #[account(
27         constraint = from_earn_manager_account.is_active @ ExtError::NotActive,
28         seeds = [EARN_MANAGER_SEED, signer.key().as_ref()],
29         bump = from_earn_manager_account.bump,
30     )]
31     pub from_earn_manager_account: Account<'info, EarnManager>,
32
33     #[account(
34         constraint = to_earn_manager_account.is_active @ ExtError::NotActive,
35         seeds = [EARN_MANAGER_SEED, to_earn_manager.as_ref()],
36         bump = to_earn_manager_account.bump,
37     )]
38     pub to_earn_manager_account: Account<'info, EarnManager>,
39 }
```

BVSS

AO:S/AC:L/AX:L/R:P/S:U/C:N/A:M/I:C/D:C/Y:C (1.6)

Recommendation

To address this issue, it is recommended to authorize an additional active authority to transfer earner accounts associated with a removed earn manager to another active manager.

Remediation Comment

ACKNOWLEDGED: The **MO team** acknowledged the risk and confirms this behavior is by design. Earners tied to an inactive Earn Manager are considered orphaned and can be removed, after which they may be added by another manager. This avoids the need to re-enable a removed manager, though it may result in yield loss if the earner had pending rewards.

7.8 THE PROGRAM RELIES ON OFF-CHAIN LOGIC

// INFORMATIONAL

Description

In certain cases, the protocol relies on off-chain logic and assumes that specific instructions will only be invoked by a predefined, trusted authority. However, the program does not enforce validation of the input arguments, relying instead on the caller to provide them correctly.

- The `remove_registrar_earner` instruction uses the `verify_not_in_tree` function to ensure that a given earner is no longer part of the Merkle tree. However, this function assumes that the items in the Merkle tree are sorted. If this assumption is not met, the algorithm may not behave as intended and could incorrectly conclude that an existing earner is no longer in the tree.
- Both `earn::claim_for` and `earn_ext::claim_for` instructions rely on the `snapshot_balance` argument provided by the caller. However, this value is not validated by the program. If the calling authority supplies an incorrect `snapshot_balance`, it could lead to reward distributions that are either higher or lower than what the recipient is actually eligible for.
- Critical program operations—such as reward distribution, adding or removing earners, and configuring the protocol—can only be performed by an authorized manager. As a result, earners must place their trust in this authority to carry out these actions correctly and to not abuse their privileges.

The correctness of the off-chain logic is out of scope for this audit.

BVSS

AO:S/AC:L/AX:L/R:N/S:U/C:N/A:N/I:M/D:M/Y:M (1.5)

Recommendation

To address this finding, it is recommended to provide clear and comprehensive protocol documentation that explicitly outlines all required off-chain, permissioned actions necessary for proper protocol operation. This documentation is essential not only for maintaining correct protocol behavior, but also for helping users understand how the protocol functions and what potential risks are associated with the off-chain workflow steps. Additionally, performing a security audit of all relevant workflow steps can help minimize the risk of incorrect instruction inputs and strengthen overall protocol integrity.

Remediation Comment

ACKNOWLEDGED: The **M0 team** has acknowledged this finding.

7.9 UNUSED INSTRUCTION WITH VULNERABILITY SURFACE

// INFORMATIONAL

Description

The `mark_outbox_item_as_released` instruction allows a PDA account, owned by a registered transceiver, to mark an outbox item as released without actually posting a message to the Wormhole bridge.
Currently, this instruction only sets a `released` flag specific to the transceiver program and does **not** prevent the subsequent `release_wormhole_outbound` call. As a result, it has no functional impact in the current implementation.
However, in the event of future updates to the codebase, this instruction increases the overall vulnerability surface. If misused or improperly integrated, it could allow outbox items to be incorrectly marked as released, potentially leading to inconsistencies in message handling or even loss of funds.
programs/portal/src/instructions/mark_outbox_item_as_released.rs

```
#[derive(Accounts)]
pub struct MarkOutboxItemAsReleased<'info> {
    #[account(
        seeds = [OUTBOX_ITEM_SIGNER_SEED],
        seeds::program = transceiver.transceiver_address,
        bump
    )]
    pub signer: Signer<'info>,

    pub config: NotPausedConfig<'info>,

    #[account(
        mut,
        constraint = !outbox_item.released.get(transceiver.id)? @ NTTError::MessageAlreadySent,
    )]
    pub outbox_item: Account<'info, OutboxItem>,

    #[account(
        constraint = config.enabled_transceivers.get(transceiver.id)? @ NTTError::DisabledTransceiver
    )]
    pub transceiver: Account<'info, RegisteredTransceiver>,
}

pub fn mark_outbox_item_as_released(ctx: Context<MarkOutboxItemAsReleased>) -> Result<bool> {
    let accs = ctx.accounts;
    let released = accs.outbox_item.try_release(accs.transceiver.id)?;
    Ok(released)
}
```

BVSS

AO:S/AC:L/AX:L/R:P/S:U/C:N/A:N/I:C/D:C/Y:N (1.3)

Recommendation

To address this issue, it is recommended to remove the unused `mark_outbox_item_as_released` instruction, as this instruction is not needed and only increases the vulnerability surface in future program updates.

Remediation Comment

SOLVED: The **M0 team** solved the finding by removing the `mark_outbox_item_as_released` function from the portal program.

Remediation Hash

https://github.com/m0-foundation/solana-m/pull/56

7.10 TWO-STEP EARN MANAGER AUTHORITY TRANSFER NOT IMPLEMENTED

// INFORMATIONAL

Description

The `transfer_earner` instruction allows an earn manager to transfer an earner account to a new manager. However, it does not require a signature from the new earn manager. This introduces a risk: if the earner account is accidentally transferred to a manager whose private key is compromised or inaccessible, the account may become unmanageable.

transfer_earner.rs

```
#[derive(Accounts)]
#[instruction(to_earn_manager: Pubkey)]
pub struct TransferEarner<'info> {
    pub signer: Signer<'info>,

    #[account(
        mut,
        constraint = earner_account.earn_manager == signer.key() @ ExtError::NotAuthorized,
        seeds = [EARNER_SEED, earner_account.user_token_account.as_ref()],
        bump = earner_account.bump,
    )]
    pub earner_account: Account<'info, Earner>,

    #[account(
        constraint = from_earn_manager_account.is_active @ ExtError::NotActive,
        seeds = [EARN_MANAGER_SEED, signer.key().as_ref()],
        bump = from_earn_manager_account.bump,
    )]
    pub from_earn_manager_account: Account<'info, EarnManager>,

    #[account(
        constraint = to_earn_manager_account.is_active @ ExtError::NotActive,
        seeds = [EARN_MANAGER_SEED, to_earn_manager.as_ref()],
        bump = to_earn_manager_account.bump,
    )]
    pub to_earn_manager_account: Account<'info, EarnManager>,
}
```

BVSS

AO:S/AC:L/AX:L/R:P/S:U/C:N/A:C/I:M/D:N/Y:N (1.1)

Recommendation

To address this issue, it is recommended to implement a two-step authority transfer where the new earn manager must provide its signature in order to finalize the authority transfer.

Remediation Comment

ACKNOWLEDGED: The **MO team** has acknowledged this finding.

7.11 TWO-STEP EARN AUTHORITY TRANSFER NOT IMPLEMENTED

// INFORMATIONAL

Description

The `set_earn_authority` instruction in `earn` and `ext_earn` allows setting a new earn authority for the program directly. This introduces a risk: if the earn authority is accidentally transferred to an address whose private key is compromised or inaccessible, this may lead to loss of rewards or Denial-of-Service for the earners.

set_earn_authority.rs

```
25 pub fn handler(ctx: Context<SetEarnAuthority>, new_earn_authority: Pubkey) -> Result<()> {
26     ctx.accounts.global_account.earn_authority = new_earn_authority;
27
28     Ok(())
29 }
```

BVSS

AO:S/AC:L/AX:L/R:P/S:U/C:N/A:C/I:N/D:N/Y:N (1.0)

Recommendation

To address this issue, it is recommended to implement a two-step authority transfer where the new earn authority accepts the ownership.

Remediation Comment

ACKNOWLEDGED: The **MO team** has acknowledged this finding.

8. AUTOMATED TESTING

Static Analysis Report

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.

Cargo Audit Results

ID	CRATE	DESCRIPTION
RUSTSEC-2024-0344	curve25519-dalek	Timing variability in `curve25519-dalek`'s `Scalar29::sub` / `Scalar52::sub`
RUSTSEC-2022-0093	ed25519-dalek	Double Public Key Signing Function Oracle Attack on `ed25519-dalek`
RUSTSEC-2024-0332	h2	Degradation of service in h2 servers with CONTINUATION Flood
RUSTSEC-2024-0421	idna	`idna` accepts Punycode labels that do not produce any non-ASCII when decoded
RUSTSEC-2024-0019	mio	Tokens for named pipes may be delivered after deregistration
RUSTSEC-2025-0009	ring	Some AES functions may panic when overflow checking is enabled.
RUSTSEC-2024-0336	rustls	`rustls::ConnectionCommon::complete_io` could fall into an infinite loop based on network input

---

Halborn strongly recommends conducting a follow-up assessment of the project either within six months or immediately following any material changes to the codebase, whichever comes first. This approach is crucial for maintaining the project's integrity and addressing potential vulnerabilities introduced by code modifications.