// Public Solana Program Security Assessment 07.30.2024 - 08.02.2024

Endogenous AVS

Solayer

HALBORN

Endogenous AVS - Solayer

Prepared by: HALBORN

Last Updated 08/13/2024

Date of Engagement by: July 30th, 2024 - August 2nd, 2024

Summary

100% OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
9	0	0	0	3	6

TABLE OF CONTENTS

8. Automated Testing

1. Introduction

`Solayer team` engaged `Halborn` to conduct a security assessment on their `Endogenous AVS` **Solana program** beginning on **July 30th, 2024**, and ending on **August, 5th, 2024**. The security assessment was scoped to the Solana Program provided in `endoavs-program` GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

The `Endogenous AVS` program takes the `sSOL` liquid mint and transforms it into a **synthetic asset** representing the **delegation** to a particular project, using the `delegate` instruction. These `mints` can be **undelegated** instantly if there is a need for trade, through the `undelegate` instruction.

Partners will be able to create an `endoavs account` through the `create` instruction, passing a `mint` address which they can customize. The `authority` can customize the AVS token `name`, `symbol`, `uri/url` and `metadata` of these assets through instructions. The `authority` can also transfer the `authority` to other account, which is irrevocable.

These assets use the same liquidity as the underlying `sSOL`. Ultimately, the goal is to enable `Solayer` to provide stake-weighted quality of service to the AVS.

2. Assessment Summary

Halborn was provided **6 days** for the engagement and assigned one full-time security engineer to review the security of the Solana Program 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 Endogenous AVS Solana Program.
- Ensure that the program's functionality operates as intended.

In summary, Halborn identified some low-severity and informational security issues, that were addressed and acknowledged by the Solayer team. The main ones were the following:

- System Flooding and Spamming.
- Lack of two-step authority transfer.
- Decimals should be enforced.
- Missing URI and URL prefix validation.
- Missing Metadata size validation.
- Missing Event emissions.
- Outdated dependencies.

Overall, the program in-scope is adherent to Solana's best-practices and carries consistent code quality.

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:

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
Attack Origin (AO)	Arbitrary (AO:A) Specific (AO:S)	1 0.2

EXPLOITABILIY 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: restaking-program

(b) Assessed Commit ID: 547e66a

(c) Items in scope:

- src/contexts/delegate.rs
- src/contexts/metadata.rs
- src/contexts/create.rs
- src/contexts/mod.rs
- src/contexts/manage.rs
- src/constants.rs
- src/state/endoavs.rs
- src/state/mod.rs
- src/errors.rs
- src/lib.rs

Out-of-Scope:

REMEDIATION COMMIT ID:

- d379d78
- 46c0907

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

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	3	6

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
SYSTEM FLOODING AND SPAMMING	LOW	SOLVED - 08/12/2024
MISSING URI AND URL PREFIX VALIDATION	LOW	RISK ACCEPTED
MISSING METADATA SIZE VALIDATION	LOW	RISK ACCEPTED
LACK OF TWO-STEP AUTHORITY TRANSFER	INFORMATIONAL	ACKNOWLEDGED
MISSING EVENT EMISSIONS	INFORMATIONAL	ACKNOWLEDGED
LACK OF ZERO AMOUNT VALIDATION	INFORMATIONAL	ACKNOWLEDGED
UN-SANITIZED ON-CHAIN STATE CAN BE USED AS ATTACK VECTOR	INFORMATIONAL	ACKNOWLEDGED
USE OF 'MSG!' CONSUMES ADDITIONAL COMPUTATIONAL BUDGET	INFORMATIONAL	ACKNOWLEDGED
OUTDATED DEPENDENCIES	INFORMATIONAL	SOLVED - 08/12/2024

7. FINDINGS & TECH DETAILS

7.1 SYSTEM FLOODING AND SPAMMING

// LOW

Description

The current implementation of the `endo_avs` account creation process allows for the creation of **multiple accounts with the same AVS** `name` and **does not enforce a minimum delegate amount** upon AVS's creation. This combination can be exploited by malicious actors to **flood the system** with `endo_avs` accounts that use the same `name/symbol`, for misleading and griefing purposes, effectively spamming the system with unvalid accounts.

The `endo_avs` metadata is then initialized with default values for `name`, `symbol` and `uri`. These parameters can be changed by the current `endo_avs` authority through the process described further.

– `programs/endoavs-program/src/contexts/create.rs`

```
17  #[derive(Accounts)]
18  pub struct CreateEndoAVS<'info> {
19      #[account(
20          init,
21          payer = authority,
22          seeds = [b"endo_avs", avs_token_mint.key().as_ref()],
23          bump,
24          space = 8 + EndoAVS::INIT_SPACE
25      )]
26      pub endo_avs: Account<'info, EndoAVS>,
27      #[account(mut)]
28      pub authority: Signer<'info>,
29      #[account(
30          init,
31          payer = authority,
32          mint::decimals = delegated_token_mint.decimals,
33          mint::authority = endo_avs,
34          mint::freeze_authority = endo_avs
35      )]
36      pub avs_token_mint: Box<InterfaceAccount<'info, Mint>>,
37      #[account(
38          mut,
39          address=Metadata::find_pda(&avs_token_mint.key()).0
40      )]
41      pub avs_token_metadata: UncheckedAccount<'info>,
42      #[account(
```

```
43          init_if_needed,
44          payer = authority,
45          associated_token::mint = delegated_token_mint,
46          associated_token::authority = endo_avs,
47          associated_token::token_program = token_program
48      )]
49      pub delegated_token_vault: Box<InterfaceAccount<'info, TokenAccount>>
50      #[account(
51          mint::token_program = token_program,
52          constraint = allow_as_delegated_asset(&delegated_token_mint.key()
53      )]
54      pub delegated_token_mint: Box<InterfaceAccount<'info, Mint>>,
55      pub token_program: Interface<'info, TokenInterface>,
56      pub associated_token_program: Program<'info, AssociatedToken>,
57      pub token_metadata_program: Program<'info, Metaplex>,
58      pub system_program: Program<'info, System>,
59      pub rent: Sysvar<'info, Rent>,
60  }
61
62  impl<'info> CreateEndoAVS<'info> {
63      pub fn create(&mut self, bumps: CreateEndoAVSBumps, name: String) ->
64          if name.len() > MAX_ENDO_AVS_NAME_LENGTH {
65              return Err(EndoAVSError::NameTooLong.into());
66          }
67
68          self.endo_avs.set_inner(EndoAVS {
69              name,
70              url: "".to_string(),
71              bump: bumps.endo_avs,
72              authority: self.authority.key(),
73              avs_token_mint: self.avs_token_mint.key(),
74              delegated_token_mint: self.delegated_token_mint.key(),
75              delegated_token_vault: self.delegated_token_vault.key(),
76          });
77
78          let token_metadata = DataV2 {
79              name: DEFAULT_ENDO_AVS_NAME.to_string(),
80              symbol: DEFAULT_ENDO_AVS_SYMBOL.to_string(),
81              uri: DEFAULT_ENDO_AVS_URI.to_string(),
82              seller_fee_basis_points: 0,
83              creators: None,
84              collection: None,
85              uses: None,
86          };
```

```
87
88          let bump = [self.endo_avs.bump];
89          let signer_seeds: [&[&[u8]]; 1] = [&[
90              b"endo_avs",
91              self.avs_token_mint.to_account_info().key.as_ref(),
92              &bump,
93          ][..]];
94
95          let metadata_ctx = CpiContext::new_with_signer(
96              self.token_metadata_program.to_account_info(),
97              CreateMetadataAccountsV3 {
98                  payer: self.authority.to_account_info(),
99                  update_authority: self.endo_avs.to_account_info(),
100                 mint: self.avs_token_mint.to_account_info(),
101                 metadata: self.avs_token_metadata.to_account_info(),
102                 mint_authority: self.endo_avs.to_account_info(),
103                 system_program: self.system_program.to_account_info(),
104                 rent: self.rent.to_account_info(),
105             },
106             &signer_seeds,
107         );
108
109         create_metadata_accounts_v3(metadata_ctx, token_metadata, true, t
110
111         Ok(())
112     }
113 }
```

After properly initializing an `endo_avs` account, it is possible to change its `name`, `symbol`, `uri` and `url` through the `update_token_metadata` and `update_endoavs` instructions.

– programs/endoavs-program/src/contexts/manage.rs

```
33  #[derive(Accounts)]
34  pub struct UpdateEndoAVSInfo<'info>{
35      pub authority: Signer<'info>,
36      #[account(
37          mut,
38          has_one = authority, // permission check
39          has_one = avs_token_mint,
40          seeds = [b"endo_avs", avs_token_mint.key().as_ref()],
41          bump = endo_avs.bump,
42      )]
43      pub endo_avs: Account<'info, EndoAVS>,
```

```
    pub avs_token_mint: Box<InterfaceAccount<'info, Mint>>,
    pub system_program: Program<'info, System>,
}

impl <'info> UpdateEndoAVSInfo<'info> {
    pub fn update(&mut self, name: Option<String>, url: Option<String>) -
        if let Some(name) = name {
            require!(name.len() < MAX_ENDO_AVS_NAME_LENGTH, EndoAVSError:
            self.endo_avs.name = name;
        }
        if let Some(url) = url {
            require!(url.len() < MAX_ENDO_AVS_URL_LENGTH, EndoAVSError::U
            self.endo_avs.url = url;
        }
        Ok(())
    }
}
```

- programs/endoavs-program/src/contexts/metadata.rs

```
#[derive(Accounts)]
pub struct AVSTokenMetadata<'info> {
    #[account(
        seeds = [b"endo_avs", avs_token_mint.key().as_ref()],
        bump = endo_avs.bump,
        has_one = avs_token_mint,
        has_one = authority,
    )]
    pub endo_avs: Account<'info, EndoAVS>,
    #[account(mut)]
    pub authority: Signer<'info>,
    pub avs_token_mint: Box<InterfaceAccount<'info, Mint>>,
    #[account(
        mut,
        address=Metadata::find_pda(&avs_token_mint.key()).0
    )]
    pub avs_token_metadata: UncheckedAccount<'info>,
    pub token_metadata_program: Program<'info, Metaplex>,
    pub system_program: Program<'info, System>,
    pub rent: Sysvar<'info, Rent>,
}

impl<'info> AVSTokenMetadata<'info> {
```

```rust
    pub fn update(&mut self, name: String, symbol: String, uri: String) -
        if !symbol.ends_with(REQUIRED_TOKEN_SYMBOL_SUFFIX) {
            return Err(EndoAVSError::InvalidTokenSymbol.into());
        }

        let token_metadata = DataV2 {
            name,
            symbol,
            uri,
            seller_fee_basis_points: 0,
            creators: None,
            collection: None,
            uses: None,
        };

        let bump = [self.endo_avs.bump];
        let signer_seeds: [&[&[u8]]; 1] = [&[
            b"endo_avs",
            self.avs_token_mint.to_account_info().key.as_ref(),
            &bump,
        ][..]];

        let metadata_ctx = CpiContext::new_with_signer(
            self.token_metadata_program.to_account_info(),
            anchor_spl::metadata::UpdateMetadataAccountsV2 {
                metadata: self.avs_token_metadata.to_account_info(),
                update_authority: self.endo_avs.to_account_info(),
            },
            &signer_seeds,
        );

        anchor_spl::metadata::update_metadata_accounts_v2(
            metadata_ctx,
            None,
            token_metadata.into(),
            None,
            None,
        )?;

        Ok(())
    }
}
```

During the whole cycle, there are no mechanisms in place to prevent malicious users from creating a significant high amount of dummy or fake **Endo AVS accounts**.This vulnerability has several negative consequences:

1. **System Flooding:** Malicious actors can create a large number of invalid accounts, overwhelming the system and potentially causing operational disruptions.
2. **User Confusion:** The presence of multiple spam accounts with the same symbol but invalid tokens can confuse users, leading them to interact with illegitimate accounts, which can ultimately rug legitimate users through abusing the `mint` authority. This confusion can result in permanent financial loss for users and undermine trust in the platform.
3. **Platform Legitimacy:** The proliferation of spam or invalid accounts can erode the legitimacy of the platform, as users may perceive it as unreliable or insecure.

Proof of Concept

In order to reproduce this vulnerability, the following test case can be used. It will use the same `signer` to create multiple `endo_avs` accounts with the same `name/symbol`, without delegating any amount to these newly created accounts.

PoC Code:

```
    it.only("should fail to create multiple EndoAVS with same name/symbol", as
        /// using User B in this test.
        try {
            await program.methods.create("mksSOL")
            .accounts({
                endoAvs: endo_avs_attacker,
                authority: d34db33f_account.publicKey,
                avsTokenMint: avs_token_mint_attacker.publicKey,
                avsTokenMetadata: metaplex.nfts().pdas().metadata({ mint: avs_
                delegatedTokenVault: getAssociatedTokenAddressSync(delegate_to
                delegatedTokenMint: delegate_token_mint,
                tokenProgram: TOKEN_PROGRAM_ID,
                associatedTokenProgram: ASSOCIATED_TOKEN_PROGRAM_ID,
                tokenMetadataProgram: metadata_program,
                systemProgram: SystemProgram.programId,
                rent: anchor.web3.SYSVAR_RENT_PUBKEY
            }).signers([d34db33f_account, avs_token_mint_attacker]).rpc();
            const endoavs_info = await program.account.endoAvs.fetch(endo_avs_
            assert.ok(endoavs_info);
            console.log("endoavs_info is : ", JSON.stringify(endoavs_info, nul

            await program.methods.create("mksSOL")
            .accounts({
                endoAvs: endo_avs_attacker2,
                authority: d34db33f_account.publicKey,
```

```
            avsTokenMint: avs_token_mint_attacker2.publicKey,
            avsTokenMetadata: metaplex.nfts().pdas().metadata({ mint: avs_
            delegatedTokenVault: getAssociatedTokenAddressSync(delegate_to
            delegatedTokenMint: delegate_token_mint,
            tokenProgram: TOKEN_PROGRAM_ID,
            associatedTokenProgram: ASSOCIATED_TOKEN_PROGRAM_ID,
            tokenMetadataProgram: metadata_program,
            systemProgram: SystemProgram.programId,
            rent: anchor.web3.SYSVAR_RENT_PUBKEY
        }).signers([d34db33f_account, avs_token_mint_attacker2]).rpc();
        const endoavs_info2 = await program.account.endoAvs.fetch(endo_avs
        assert.ok(endoavs_info2);
        console.log("endoavs_info2 is : ", JSON.stringify(endoavs_info2, n

        await program.methods.create("mksSOL")
        .accounts({
            endoAvs: endo_avs_attacker3,
            authority: d34db33f_account.publicKey,
            avsTokenMint: avs_token_mint_attacker3.publicKey,
            avsTokenMetadata: metaplex.nfts().pdas().metadata({ mint: avs_
            delegatedTokenVault: getAssociatedTokenAddressSync(delegate_to
            delegatedTokenMint: delegate_token_mint,
            tokenProgram: TOKEN_PROGRAM_ID,
            associatedTokenProgram: ASSOCIATED_TOKEN_PROGRAM_ID,
            tokenMetadataProgram: metadata_program,
            systemProgram: SystemProgram.programId,
            rent: anchor.web3.SYSVAR_RENT_PUBKEY
        }).signers([d34db33f_account, avs_token_mint_attacker3]).rpc();
        const endoavs_info3 = await program.account.endoAvs.fetch(endo_avs
        assert.ok(endoavs_info3);
        console.log("endoavs_info3 is : ", JSON.stringify(endoavs_info3, n

    } catch (error) {
        assert(error.message);
        console.error(error);
    }
});
```

Stack traces:

```
  endoavs-program::
endoavs_info is :  {

```

```
  "bump": 255,
  "authority": "FYTjAm73BmkAFDm9VqBAtsryAVkjUPge9EG7HSNjcqeq",
  "avsTokenMint": "EzByfLuvkTaRdKSoafLGEZX2Pw2e433dx1Kuat3FcEsT",
  "delegatedTokenMint": "sSo14endRuUbvQaJS3dq36Q829a3A6BEfoeeRGJywEh",
  "delegatedTokenVault": "CSiRqvu3nvzH4NQHidWt11wgwz8WMUS1A5JSMEGWD6Lq",
  "name": "mksSOL",
  "url": ""
}
endoavs_info2 is :  {
  "bump": 253,
  "authority": "FYTjAm73BmkAFDm9VqBAtsryAVkjUPge9EG7HSNjcqeq",
  "avsTokenMint": "5pqkYdEh5xpsiaRBNxs6LTnsswGrw4U7NzxMRhqnGFKW",
  "delegatedTokenMint": "sSo14endRuUbvQaJS3dq36Q829a3A6BEfoeeRGJywEh",
  "delegatedTokenVault": "135jLTgYrmMivNF2Eqx5qCYBp7eSvGRkYsmiM4tTFzeX",
  "name": "mksSOL",
  "url": ""
}
endoavs_info3 is :  {
  "bump": 255,
  "authority": "FYTjAm73BmkAFDm9VqBAtsryAVkjUPge9EG7HSNjcqeq",
  "avsTokenMint": "25LkCus9gAquyDMnpupDoqWgUqosCUiQM9s6rU3coKGC",
  "delegatedTokenMint": "sSo14endRuUbvQaJS3dq36Q829a3A6BEfoeeRGJywEh",
  "delegatedTokenVault": "ED25CKR64L9gWo9Uvt71V64ZHuBEFsETUv4BMq4XzPtT",
  "name": "mksSOL",
  "url": ""
}
```

Execution:

```
  endoavs-program::
endoavs_info is :  {
  "bump": 255,
  "authority": "FYTjAm73BmkAFDm9VqBAtsryAVkjUPge9EG7HSNjcqeq",
  "avsTokenMint": "EzByfLuvkTaRdKSoafLGEZX2Pw2e433dx1Kuat3FcEsT",
  "delegatedTokenMint": "sSo14endRuUbvQaJS3dq36Q829a3A6BEfoeeRGJywEh",
  "delegatedTokenVault": "CSiRqvu3nvzH4NQHidWt11wgwz8WMUS1A5JSMEGWD6Lq",
  "name": "mksSOL",
  "url": ""
}
endoavs_info2 is :  {
  "bump": 253,
  "authority": "FYTjAm73BmkAFDm9VqBAtsryAVkjUPge9EG7HSNjcqeq",
  "avsTokenMint": "5pqkYdEh5xpsiaRBNxs6LTnsswGrw4U7NzxMRhqnGFKW",
  "delegatedTokenMint": "sSo14endRuUbvQaJS3dq36Q829a3A6BEfoeeRGJywEh",
  "delegatedTokenVault": "135jLTgYrmMivNF2Eqx5qCYBp7eSvGRkYsmiM4tTFzeX",
  "name": "mksSOL",
  "url": ""
}
endoavs_info3 is :  {
  "bump": 255,
  "authority": "FYTjAm73BmkAFDm9VqBAtsryAVkjUPge9EG7HSNjcqeq",
  "avsTokenMint": "25LkCus9gAquyDMnpupDoqWgUqosCUiQM9s6rU3coKGC",
  "delegatedTokenMint": "sSo14endRuUbvQaJS3dq36Q829a3A6BEfoeeRGJywEh",
  "delegatedTokenVault": "ED25CKR64L9gWo9Uvt71V64ZHuBEFsETUv4BMq4XzPtT",
  "name": "mksSOL",
  "url": ""
}
```

BVSS

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

Recommendation

1. **Enforce Unique Symbols:**

Create a **PDA** `symbol_mapping` to track existing symbols, with a boolean `exists` field. This PDA will use the symbol as a seed to ensure uniqueness.

In the `create` instruction, add a `symbol_mapping` account (Program Derived Address). Use the `init` **account constraint** and the `symbol` as a seed for the PDA derivation. This will effectively block duplicate symbols from being used to create new `endo_avs` accounts, preventing system flooding. Additionally, the `create` function should set the `exists` field of the provided PDA to `true`.

This approach ensures that each `endo_avs` account has a unique `symbol`, mitigating the risk of system flooding and maintaining the integrity of the platform.

2. Enforce a Minimum delegation amount upon `endo_avs` creation:

Implement a **minimum delegation** `amount` requirement upon the creation of `endo_avs` accounts. This will discourage malicious users from creating numerous low-value accounts, as they will have no financial incentive and will incur a direct loss of `sSOL`.

By enforcing a minimum delegation amount, you can deter malicious actors from flooding the system with invalid accounts, thereby enhancing the security and reliability of the platform.

Remediation Plan

SOLVED: The **Solayer team** has solved this issue by enforcing a minimum deposit fee. The commit hash containing the modifications is `d379d7898a98d4403f8305896bc3faf7e162cf44`.

Remediation Hash

https://github.com/solayer-labs/restaking-program/commit/d379d7898a98d4403f8305896bc3faf7e162cf44

7.2 MISSING URI AND URL PREFIX VALIDATION

// LOW

Description

The `authority` of each `endo_avs` account is entitled to change the `name` and the `url` through the `update` method, as follows:

– `programs/endoavs-program/src/contexts/manage.rs`

```
48  impl <'info> UpdateEndoAVSInfo<'info> {
49      pub fn update(&mut self, name: Option<String>, url: Option<String>) -
50          if let Some(name) = name {
51              require!(name.len() < MAX_ENDO_AVS_NAME_LENGTH, EndoAVSError:
52              self.endo_avs.name = name;
53          }
54          if let Some(url) = url {
55              require!(url.len() < MAX_ENDO_AVS_URL_LENGTH, EndoAVSError::U
56              self.endo_avs.url = url;
57          }
58          Ok(())
59      }
```

When updating the metadata, this verification is also missing.

– `programs/endoavs-program/src/contexts/metadata.rs`

```
37  impl<'info> AVSTokenMetadata<'info> {
38      pub fn update(&mut self, name: String, symbol: String, uri: String) -
39          if !symbol.ends_with(REQUIRED_TOKEN_SYMBOL_SUFFIX) {
40              return Err(EndoAVSError::InvalidTokenSymbol.into());
41          }
42
43          let token_metadata = DataV2 {
44              name,
45              symbol,
46              uri,
47              seller_fee_basis_points: 0,
48              creators: None,
49              collection: None,
50              uses: None,
51          };
52
53
```

```
        let bump = [self.endo_avs.bump];
        let signer_seeds: [&[&[u8]]; 1] = [&[
            b"endo_avs",
            self.avs_token_mint.to_account_info().key.as_ref(),
            &bump,
        ][..]];

        let metadata_ctx = CpiContext::new_with_signer(
            self.token_metadata_program.to_account_info(),
            anchor_spl::metadata::UpdateMetadataAccountsV2 {
                metadata: self.avs_token_metadata.to_account_info(),
                update_authority: self.endo_avs.to_account_info(),
            },
            &signer_seeds,
        );

        anchor_spl::metadata::update_metadata_accounts_v2(
            metadata_ctx,
            None,
            token_metadata.into(),
            None,
            None,
        )?;

        Ok(())
    }
}
```

There are no verifications in place whether the provided `url` or `uri` starts with an expected format, such as `https://`, what could lead to unintended behavior in off-chain premises and also pollute the account with inaccurate information.

BVSS

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

Recommendation

It is recommended to add simple verifications to check whether the provided `uri` and `url` prefixes matches pre-determined formats.

Remediation Plan

RISK ACCEPTED: The **Solayer team** has accepted the risk related to this finding.

7.3 MISSING METADATA SIZE VALIDATION

// LOW

Description

The `authority` account of each `endo_avs` is entitled to update its metadata information, including `name`, `symbol` and `uri`, as strings. However, there are no verifications in place to prevent those user-provided inputs from being excessively large.

– `programs/endoavs-program/src/contexts/metadata.rs`

```
37  impl<'info> AVSTokenMetadata<'info> {
38      pub fn update(&mut self, name: String, symbol: String, uri: String) -
39          if !symbol.ends_with(REQUIRED_TOKEN_SYMBOL_SUFFIX) {
40              return Err(EndoAVSError::InvalidTokenSymbol.into());
41          }
42
43          let token_metadata = DataV2 {
44              name,
45              symbol,
46              uri,
47              seller_fee_basis_points: 0,
48              creators: None,
49              collection: None,
50              uses: None,
51          };
52
53          let bump = [self.endo_avs.bump];
54          let signer_seeds: [&[&[u8]]; 1] = [&[
55              b"endo_avs",
56              self.avs_token_mint.to_account_info().key.as_ref(),
57              &bump,
58          ][..]];
59
60          let metadata_ctx = CpiContext::new_with_signer(
61              self.token_metadata_program.to_account_info(),
62              anchor_spl::metadata::UpdateMetadataAccountsV2 {
63                  metadata: self.avs_token_metadata.to_account_info(),
64                  update_authority: self.endo_avs.to_account_info(),
65              },
66              &signer_seeds,
67          );
68
69
```

```
            anchor_spl::metadata::update_metadata_accounts_v2(
                metadata_ctx,
                None,
                token_metadata.into(),
                None,
                None,
            )?;

            Ok(())
        }
    }
```

The lack of validation of user-provided inputs for excessively large values can lead to unintended behavior in off-chain premises, such as weird website rendering, and also pollute the system state with inadequate data.

BVSS

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

Recommendation

It is recommended to check the length of user-provided inputs against a safe threshold.

Remediation Plan

RISK ACCEPTED: The **Solayer team** has accepted the risk related to this finding.

7.4 LACK OF TWO-STEP AUTHORITY TRANSFER

// INFORMATIONAL

Description

The `endoavs` program in-scope allows the current `authority` account of each Endogenous AVS to transfer the `authority` to another account. Such action is permanent and irrevocable.
The existing implementation of the `transfer_authority` instruction employs a one-step procedure for authority delegation, which presents a security concern. This method lacks a safeguard against inadvertent delegations or potential hostile takeovers.
– `programs/endoavs-program/src/contexts/manage.rs`

```
25 impl<'info> TransferAuthority<'info> {
26     pub fn transfer_authority(&mut self) -> Result<()> {
27         self.endo_avs.authority = self.new_authority.key();
28         msg!("Transferred authority to {}", self.new_authority.key());
29         Ok(())
30     }
31 }
```

BVSS

AO:S/AC:L/AX:L/C:N/I:H/A:N/D:N/Y:N/R:N/S:C (1.9)

Recommendation

To resolve this issue, it is advisable to establish a two-step process for `authority` transfer, thereby enhancing the security of the operation. The current `authority` would first propose a new candidate `authority`, who would then need to formally **accept** the role.
This process would be structured as follows:
1. **Proposal by Current Authority:** The current signer proposes a new candidate signer. This action updates the `candidate_authority` field in the `endo_avs` account's state. This step assumes the prior creation of an additional field `candidate_authority` on `state/endoavs.rs`.
2. **Acceptance by New Authority:** The proposed `candidate_authority` formally **accepts the role**. This step transfers the `authority` status from the current `authority` to the `candidate_authority`.

Remediation Plan

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

7.5 MISSING EVENT EMISSIONS

// INFORMATIONAL

Description

It is considered best practice when developing Solana programs to emit events when important modifications to the state are performed, such as **Metadata modifications and** `authority` **transfers**.

– programs/endoavs-program/src/contexts/manage.rs

```
impl<'info> TransferAuthority<'info> {
    pub fn transfer_authority(&mut self) -> Result<()> {
        self.endo_avs.authority = self.new_authority.key();
        msg!("Transferred authority to {}", self.new_authority.key());
        Ok(())
    }
}
```

– programs/endoavs-program/src/contexts/manage.rs

```
impl <'info> UpdateEndoAVSInfo<'info> {
    pub fn update(&mut self, name: Option<String>, url: Option<String>) -
        if let Some(name) = name {
            require!(name.len() < MAX_ENDO_AVS_NAME_LENGTH, EndoAVSError:
            self.endo_avs.name = name;
        }
        if let Some(url) = url {
            require!(url.len() < MAX_ENDO_AVS_URL_LENGTH, EndoAVSError::U
            self.endo_avs.url = url;
        }
        Ok(())
    }
}
```

– programs/endoavs-program/src/contexts/metadata.rs

```
impl<'info> AVSTokenMetadata<'info> {
    pub fn update(&mut self, name: String, symbol: String, uri: String) -
        if !symbol.ends_with(REQUIRED_TOKEN_SYMBOL_SUFFIX) {
            return Err(EndoAVSError::InvalidTokenSymbol.into());
        }
```

```
        let token_metadata = DataV2 {
            name,
            symbol,
            uri,
            seller_fee_basis_points: 0,
            creators: None,
            collection: None,
            uses: None,
        };

        let bump = [self.endo_avs.bump];
        let signer_seeds: [&[&[u8]]; 1] = [&[
            b"endo_avs",
            self.avs_token_mint.to_account_info().key.as_ref(),
            &bump,
        ][..]];

        let metadata_ctx = CpiContext::new_with_signer(
            self.token_metadata_program.to_account_info(),
            anchor_spl::metadata::UpdateMetadataAccountsV2 {
                metadata: self.avs_token_metadata.to_account_info(),
                update_authority: self.endo_avs.to_account_info(),
            },
            &signer_seeds,
        );

        anchor_spl::metadata::update_metadata_accounts_v2(
            metadata_ctx,
            None,
            token_metadata.into(),
            None,
            None,
        )?;

        Ok(())
    }
}
```

It was identified that **events are not being emitted** for the annotated important state operations.

BVSS

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

Recommendation

Ensure that all critical actions within the program emit corresponding events, such as when transferring the authority or updating token information.

Remediation Plan

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

7.6 LACK OF ZERO AMOUNT VALIDATION

// INFORMATIONAL

Description

The program in-scope does not prevent the `delegate` and `undelegate` methods from being called with `amount == 0`.

– programs/endoavs-program/src/contexts/delegate.rs

```
87      pub fn delegate(&mut self, amount: u64) -> Result<()> {
88          // Transfer tokens from user to the delegated token vault
89          let transfer_accounts = TransferChecked {
90              from: self.staker_delegated_token_account.to_account_info(),
91              to: self.delegated_token_vault.to_account_info(),
92              mint: self.delegated_token_mint.to_account_info(),
93              authority: self.staker.to_account_info(),
94          };
95          let transfer_ctx = CpiContext::new(self.token_program.to_account_
96          transfer_checked(transfer_ctx, amount, self.delegated_token_mint.
97
98          let bump = [self.endo_avs.bump];
99          let signer_seeds: [&[&[u8]]; 1] = [&[
100             b"endo_avs",
101             self.avs_token_mint.to_account_info().key.as_ref(),
102             &bump,
103         ][..]];
104
105         let mint_ctx = CpiContext::new_with_signer(
106             self.token_program.to_account_info(),
107             MintTo {
108                 to: self.staker_avs_token_account.to_account_info(),
109                 mint: self.avs_token_mint.to_account_info(),
110                 authority: self.endo_avs.to_account_info(),
111             },
112             &signer_seeds[..],
113         );
114         mint_to(mint_ctx, amount)?;
115
116         Ok(())
117     }
```

– programs/endoavs-program/src/contexts/delegate.rs

```rust
    pub fn undelegate(&mut self, amount: u64) -> Result<()> {
        // Burn EndoAVS tokens from the user
        let burn_accounts = Burn {
            from: self.staker_avs_token_account.to_account_info(),
            mint: self.avs_token_mint.to_account_info(),
            authority: self.staker.to_account_info(),
        };
        let burn_ctx = CpiContext::new(self.token_program.to_account_info
        burn(burn_ctx, amount)?;

        let bump = [self.endo_avs.bump];
        let signer_seeds: [&[&[u8]]; 1] = [&[
            b"endo_avs",
            self.avs_token_mint.to_account_info().key.as_ref(),
            &bump,
        ][..]];

        let transfer_accounts = TransferChecked {
            from: self.delegated_token_vault.to_account_info(),
            to: self.staker_delegated_token_account.to_account_info(),
            mint: self.delegated_token_mint.to_account_info(),
            authority: self.endo_avs.to_account_info(),
        };
        let transfer_ctx = CpiContext::new_with_signer(
            self.token_program.to_account_info(),
            transfer_accounts,
            &signer_seeds,
        );
        transfer_checked(transfer_ctx, amount, self.delegated_token_mint.

        Ok(())
    }
```

While this condition does not lead to immediate financial loss, it should be checked to keep overall consistency.

Score

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation

Consider adding a verification before the execution of the `delegate` and `undelegate` methods, blocking those actions with `amount == 0`.

Remediation Plan

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

7.7 UN-SANITIZED ON-CHAIN STATE CAN BE USED AS ATTACK VECTOR

// INFORMATIONAL

Description

The current implementation of methods that change `endo_avs` token informations lacks proper input validation, which can lead to security vulnerabilities.
Metadata, such as `name`, `symbol`, and `uri`, and also token `name` and `symbol`, can be freely provided by users when creating or updating EndoAVS metadata. If not properly sanitized in the front-end, this metadata can be used as an attack vector for Stored Cross-Site Scripting (XSS), and other well-known web vulnerabilities.
– `programs/endoavs-program/src/contexts/manage.rs`

```
48  impl <'info> UpdateEndoAVSInfo<'info> {
49      pub fn update(&mut self, name: Option<String>, url: Option<String>) -
50          if let Some(name) = name {
51              require!(name.len() < MAX_ENDO_AVS_NAME_LENGTH, EndoAVSError:
52              self.endo_avs.name = name;
53          }
54          if let Some(url) = url {
55              require!(url.len() < MAX_ENDO_AVS_URL_LENGTH, EndoAVSError::U
56              self.endo_avs.url = url;
57          }
58          Ok(())
59      }
```

```
37  impl<'info> AVSTokenMetadata<'info> {
38      pub fn update(&mut self, name: String, symbol: String, uri: String) -
39          if !symbol.ends_with(REQUIRED_TOKEN_SYMBOL_SUFFIX) {
40              return Err(EndoAVSError::InvalidTokenSymbol.into());
41          }
42
43          let token_metadata = DataV2 {
44              name,
45              symbol,
46              uri,
47              seller_fee_basis_points: 0,
48              creators: None,
49              collection: None,
50              uses: None,
51
```

```
        };

        let bump = [self.endo_avs.bump];
        let signer_seeds: [&[&[u8]]; 1] = [&[
            b"endo_avs",
            self.avs_token_mint.to_account_info().key.as_ref(),
            &bump,
        ][..]];

        let metadata_ctx = CpiContext::new_with_signer(
            self.token_metadata_program.to_account_info(),
            anchor_spl::metadata::UpdateMetadataAccountsV2 {
                metadata: self.avs_token_metadata.to_account_info(),
                update_authority: self.endo_avs.to_account_info(),
            },
            &signer_seeds,
        );

        anchor_spl::metadata::update_metadata_accounts_v2(
            metadata_ctx,
            None,
            token_metadata.into(),
            None,
            None,
        )?;

        Ok(())
    }
}
```

If an attacker manages to successfully craft valid payloads using on-chain state to weaponize it, the attack can ultimately lead to critical consequences such as account take-over and arbitrary script execution on victim's browser.

Score

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation

Given that on-chain primitives do not allow for proper sanitization of inputs, and considering the myriad of payloads that can affect web applications, it is recommended to conduct a thorough examination of off-chain components, such as back-end and front-end applications.

Ensure that malicious on-chain payloads are properly sanitized in off-chain premises to prevent payloads from being rendered in a way that could execute arbitrary code in users' browsers.

Remediation Plan

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

7.8 USE OF 'MSG!' CONSUMES ADDITIONAL COMPUTATIONAL BUDGET

// INFORMATIONAL

Description

The usage of `msg!` is usually advisable during tests, and will incur in additional computational budget when the instruction is processed in Mainnet.

– programs/endoavs-program/src/contexts/manage.rs

```
25 impl<'info> TransferAuthority<'info> {
26     pub fn transfer_authority(&mut self) -> Result<()> {
27         self.endo_avs.authority = self.new_authority.key();
28         msg!("Transferred authority to {}", self.new_authority.key());
29         Ok(())
30     }
31 }
```

Score

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation

Consider removing debugging messages before Mainnet deployment.

Remediation Plan

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

7.9 OUTDATED DEPENDENCIES

// INFORMATIONAL

Description

It was identifying during the assessment of the program `endo_avs` in-scope that its dependencies for the Anchor framework and also for Solana are not current.

```
[[package]]
name = "solana-program"
version = "1.18.7"
```

```
[[package]]
name = "anchor-lang"
version = "0.29.0"
```

Score

AO:A/AC:L/AX:L/C:N/I:N/A:N/D:N/Y:N/R:N/S:C (0.0)

Recommendation

It is recommended to update dependencies to their current versions, as specified:

- Solana: `v1.18.20`
- Anchor: `v0.31.0`

Remediation Plan

SOLVED: The **Solayer team** has solved this issue as recommended. The commit hash containing the modification is `46c09073a6dad390f435dc76f17e35849f2c6d1b`.

Remediation Hash

https://github.com/solayer-labs/restaking-program/commit/46c09073a6dad390f435dc76f17e35849f2c6d1b

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	DESCCRIPTION
RUSTSEC-2022-0093	ed25519-dalek	Double Public Key Signing Function Oracle Attack on `ed255109-dalek`
RUSTSEC-2024-0344	curve25519-dalek	Timing variability in `curve25519-dalek`'s Scalar29::sub/Scalar52::sub
RUSTSEC-2021-0145	atty	Potential unaligned read

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.