// Private Solana Program Security Assessment 04.15.2024 - 05.22.2024

Sub Registrar

Bonfida

HALBORN

Sub Registrar - Bonfida

Prepared by: HALBORN

Last Updated 06/06/2024

Date of Engagement by: April 15th, 2024 - May 22nd, 2024

Summary

100% ⓘ OF ALL REPORTED FINDINGS HAVE BEEN ADDRESSED

ALL FINDINGS	CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
8	1	0	1	2	4

TABLE OF CONTENTS

1. Introduction

`Bonfida team` engaged `Halborn` to conduct a security assessment on their **sub-registrar Solana program** beginning on April 15th, 2024 and ending on May 22nd, 2024. The security assessment was scoped to the Solana Programs provided in [sub-registrar](https://github.com/Bonfida/sub-registrar/tree/master) GitHub repository. Commit hashes and further details can be found in the Scope section of this report.

The `SubRegistrar` program empowers owners of .sol domains to establish a subdomain registrar, streamlining the process of issuing subdomains. Subregistrar owners have the flexibility to craft a bespoke pricing structure based on the length of the subdomain, choose their preferred token mint, and restrict issuance solely to holders of a specific NFT collection.

2. Assessment Summary

`Halborn` was provided 5.5 weeks 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 improvements to reduce the likelihood and impact of multiple risks, which has been partially addressed by the `Bonfida team`. The main ones were the following:

- Unauthorized subdomain revocation
- Loss of lamports by the owner when admin revokes
- Subreverse account check missing in admin register
- NFT gated collection check missing

3.

4. 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 (ècargo audit)̀.

- Local runtime testing (ès̀olana-test-framework)̀

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
Attack Cost (AC)	Low (AC:L) Medium (AC:M) High (AC:H)	1 0.67 0.33

EXPLOITABILIY METRIC (M_E)	METRIC VALUE	NUMERICAL VALUE
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

IMPACT METRIC (M_I)	METRIC VALUE	NUMERICAL VALUE
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
Informational	0 - 1.9

5. SCOPE

FILES AND REPOSITORY

(a) Repository: sub-registrar

(b) Assessed Commit ID: de7aff8

(c) Items in scope:

- /program/src/processor/admin_register.rs
- /program/src/processor/admin_revoke.rs
- /program/src/processor/close_registrar.rs
- /program/src/processor/create_registrar.rs
- /program/src/processor/delete_subdomain_record.rs
- /program/src/processor/edit_registrar.rs
- /program/src/processor/nft_owner_revoke.rs
- /program/src/processor/register.rs
- /program/src/processor/unregister.rs
- /program/src/state/mint_record.rs
- /program/src/state/registry.rs
- /program/src/state/schedule.rs
- /program/src/state/subdomain_record.rs
- /program/src/cpi.rs
- /program/src/entrypoint.rs
- /program/src/error.rs
- /program/src/instruction.rs
- /program/src/lib.rs
- /program/src/processor.rs
- /program/src/revoke_unchecked.rs
- /program/src/state.rs
- /program/src/utils.rs

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

REMEDIATION COMMIT ID:

- 5dad2445dad244
- 58151845815184
- 230dfcb230dfcb
- e14ab1be14ab1b

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

6. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
1	0	1	2	4

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
UNAUTHORIZED SUBDOMAIN REVOCATION	Critical	SOLVED - 04/26/2024
LOSS OF LAMPORTS BY THE OWNER WHEN THE ADMIN REVOKES	Medium	SOLVED - 05/22/2024
NFT GATED COLLECTION CHECK MISSING	Low	RISK ACCEPTED
SUBREVERSE ACCOUNT CHECK MISSING IN ADMIN REGISTER	Low	SOLVED - 05/22/2024
HARDCODED FEE ADDRESS IN REGISTER	Informational	ACKNOWLEDGED
AUTHORITY ADDRESS CHECK MISSING	Informational	ACKNOWLEDGED
POTENTIAL SUBDOMAIN IMPERSONATION	Informational	SOLVED - 05/26/2024
TOKEN MINT AND FEE ACCOUNT CHECK MISSING	Informational	ACKNOWLEDGED

7. FINDINGS & TECH DETAILS

7.1 UNAUTHORIZED SUBDOMAIN REVOCATION

// CRITICAL

Description

The `CreateRegistrar` instruction empowers a domain owner to establish a Registrar by furnishing various accounts and parameter values, among which is the optional **nft_gated_collection**. This parameter serves to restrict access to records behind an NFT collection. When provided, it enables users possessing NFTs associated with that collection to register subdomains. Subsequently, the MintRecord state tracks the number of subdomains minted through the given NFT.

The `NftOwnerRevoke` instruction enables the owner of the NFT to revoke previously registered subdomains. However, it neglects to verify whether the mint of the provided NFT account corresponds to the mint of the mint_record account. Consequently, any user with a verified NFT linked to **nft_gated_collection** could potentially revoke other users' subdomains by accessing the lamports allocated to those locks.

nft_owner_revoke.rs

```
115     let sub_record =
116 SubDomainRecord::from_account_info(accounts.sub_record,
117 Tag::SubRecord)?;
118     let registrar = Registrar::from_account_info(accounts.registrar,
119 Tag::Registrar)?;
120     let mint_record =
121 MintRecord::from_account_info(accounts.nft_mint_record,
122 Tag::MintRecord)?;
123     let collection = registrar
124         .nft_gated_collection
125         .ok_or(SubRegisterError::MustHaveCollection)?;
126
127     let mint = check_nft_holding_and_get_mint(accounts.nft_account,
128 accounts.nft_owner.key)?;
129
130    check_metadata(accounts.nft_metadata, &collection)?;
131
132     let (pda, _) = Metadata::find_pda(&mint);
133     msg!("pda: {}", pda);
134
135
136     check_account_key(accounts.nft_metadata, &pda)?;
137
138     let (subrecord_key, _) =
139 SubDomainRecord::find_key(accounts.sub_domain_account.key,
140
```

```
141 program_id);
142
143     check_account_key(accounts.sub_record, &subrecord_key)?;
144     check_account_key(accounts.parent_domain,
145 &registrar.domain_account)?;
146
147     if let Some(sub_mint_rec) = sub_record.mint_record {
148         check_account_key(accounts.nft_mint_record, &sub_mint_rec)?;
149     } else {
150         return Err(SubRegisterError::WrongMintRecord.into());
151     }
152
153     revoke_unchecked::revoke_unchecked(
154         registrar,
            sub_record,
            Some(mint_record),
            accounts.registrar,
            accounts.sub_domain_account,
            accounts.parent_domain,
            accounts.name_class,
            accounts.spl_name_service,
            accounts.sub_record,
            accounts.nft_owner,
            Some(accounts.nft_mint_record),
```

Proof of Concept

1. A registrar is created with gate nft collection
2. Bob register with a verified nft for that collection
3. Alice revokes Bob's subdomain calling NftOwnerRevoke providing its verfied nft for that collection but which is different than Bob's
4. Check the Subdomain is revoked and the mint record count has been decreased.

```
registrar Data: Registrar { tag: Registrar, nonce: 255, authority: 11111111111111111111111111111111, fee_account: 7WdrT3PETDZUDyhUi7Gp8WChGDdqSnAavX6VH1MSpYkS, mint: 1111111ogCyDbaRMvkdsHB3qfdyFYaG1WtRUAf
dh, domain_account: GaZRomHxg8aqZ31PDvpABoeFPyYPWynUQt9G5V4FkKFj, total_sub_created: 0, nft_gated_collection: Some(E5ZnBpH9DYcxRkumKdS4ayJ3Ftb6o3E8wSbXw4N92GWg), max_nft_mint: 2, allow_revoke: false, pric
e_schedule: [Price { length: 1, price: 10000000 }, Price { length: 2, price: 10000000 }] }
[*]Bob Register
[2024-05-21T18:20:23.520543000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs invoke [1]
[2024-05-21T18:20:23.521788000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.521799000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.521807000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.521813000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+] Instruction: Register Instruction
[2024-05-21T18:20:23.523342000Z DEBUG solana_runtime::message_processor::stable_log] Program log: collection verified
[2024-05-21T18:20:23.523353000Z DEBUG solana_runtime::message_processor::stable_log] Program log: collection key checke
[2024-05-21T18:20:23.523835000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [2]
[2024-05-21T18:20:23.523856000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.524361000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Get Domain Price..
[2024-05-21T18:20:23.524690000Z DEBUG solana_runtime::message_processor::stable_log] Program log: len: 10
[2024-05-21T18:20:23.524769000Z DEBUG solana_runtime::message_processor::stable_log] Program log: price index: 1
[2024-05-21T18:20:23.525059000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-05-21T18:20:23.525430000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
[2024-05-21T18:20:23.526092000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4645 of 171325 compute units
[2024-05-21T18:20:23.526114000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-05-21T18:20:23.526433000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-05-21T18:20:23.526736000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
[2024-05-21T18:20:23.527385000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4645 of 164316 compute units
[2024-05-21T18:20:23.527406000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-05-21T18:20:23.528026000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [2]
[2024-05-21T18:20:23.528432000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.528443000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.528495000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.528502000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Create
[2024-05-21T18:20:23.529181000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [3]
[2024-05-21T18:20:23.529197000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.529428000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [3]
[2024-05-21T18:20:23.529444000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.529759000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [3]
[2024-05-21T18:20:23.529775000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.529918000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 12014 of 155702 compute units
[2024-05-21T18:20:23.529936000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:20:23.533343000Z DEBUG solana_runtime::message_processor::stable_log] Program jCebN34bUfdeUYJT13J1yG16XWQpt5PDx6Mse9GUqhR invoke [2]
[2024-05-21T18:20:23.534248000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.534261000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.534270000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.534276000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: CreateReverse
[2024-05-21T18:20:23.540791000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [3]
[2024-05-21T18:20:23.541318000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.541337000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.541394000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.541402000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Create
[2024-05-21T18:20:23.542108000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [4]
[2024-05-21T18:20:23.542148000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.542394000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [4]
[2024-05-21T18:20:23.542416000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.542739000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [4]
[2024-05-21T18:20:23.542765000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.542917000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 12095 of 80512 compute units
[2024-05-21T18:20:23.542941000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:20:23.543549000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [3]
[2024-05-21T18:20:23.543834000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.543845000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.543908000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.543915000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Update Data
```

```
[2024-05-21T18:20:23.543834000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.543845000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.543908000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.543915000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Update Data
[2024-05-21T18:20:23.544149000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 2474 of 64812 compute units
[2024-05-21T18:20:23.544166000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:20:23.544465000Z DEBUG solana_runtime::message_processor::stable_log] Program jCebN34bUfdeUYJT13J1yG16XWQpt5PDx6Mse9GUqhR consumed 58852 of 120131 compute units
[2024-05-21T18:20:23.544485000Z DEBUG solana_runtime::message_processor::stable_log] Program jCebN34bUfdeUYJT13J1yG16XWQpt5PDx6Mse9GUqhR success
[2024-05-21T18:20:23.544884000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [2]
[2024-05-21T18:20:23.544903000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:20:23.545194000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs consumed 143026 of 200000 compute units
[2024-05-21T18:20:23.545220000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs success
Subrecord Data: SubDomainRecord { tag: SubRecord, registrar: BFwZ4WpwqQzNCypqnu1qiGoNCBiY2oru44a2fy4HZJe7, sub_key: 3GswMaHBB4JNxgYNXSYKMX4oy3itniz9izawBwMCBPBA, mint_record: Some(2GPDAdPZYEMYa4b4D2thCfqo
xDi9gbrACwVb6p4EcNRd) }
Mintrecord Data: MintRecord { tag: MintRecord, count: 1, mint: BM6UwALQkF5yMv85Nv5YiVNVs5wgiRX2qTRPGbCSbvfF }
[*]Alice Owner NFT Revoke
nft mint: 1111111ogCyDbaRMvkdsHB3qfdyFYaG1WtRUAfdh
[2024-05-21T18:20:23.551607000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs invoke [1]
[2024-05-21T18:20:23.552344000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.552352000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.552361000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.552367000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+] Instruction: NFT owner revoke instruction
[2024-05-21T18:20:23.553722000Z DEBUG solana_runtime::message_processor::stable_log] Program log: collection verified
[2024-05-21T18:20:23.553732000Z DEBUG solana_runtime::message_processor::stable_log] Program log: collection key checke
[2024-05-21T18:20:23.555642000Z DEBUG solana_runtime::message_processor::stable_log] Program log: pda: Bi4ncahFxNsy4cjmsWmDy3w6LoW9HbRH6kQRywnC1mtj
[2024-05-21T18:20:23.557740000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Name Record Header owner del sub domian: 2eYD8Y5ybZffK4VeJj5NQQegTzGouQnTVYgEAvyXoc3t
[2024-05-21T18:20:23.557748000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+]Transfer Subdomain...
[2024-05-21T18:20:23.558119000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [2]
[2024-05-21T18:20:23.558457000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.558469000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.558488000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.558494000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer Ownership
[2024-05-21T18:20:23.558750000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 2517 of 158976 compute units
[2024-05-21T18:20:23.558768000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:20:23.560754000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Name Record Header owner del sub domian: BFwZ4WpwqQzNCypqnu1qiGoNCBiY2oru44a2fy4HZJe7
[2024-05-21T18:20:23.560769000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+]Delete..
[2024-05-21T18:20:23.561015000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [2]
[2024-05-21T18:20:23.561266000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:20:23.561277000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:20:23.561289000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:20:23.561295000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Delete Name
[2024-05-21T18:20:23.561467000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 1847 of 141509 compute units
[2024-05-21T18:20:23.561485000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:20:23.562095000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Name Record Header owner del sub domian: 11111111111111111111111111111111
[2024-05-21T18:20:23.562385000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs consumed 65366 of 200000 compute units
[2024-05-21T18:20:23.562405000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs success
registrar Data: Registrar { tag: Registrar, nonce: 255, authority: 11111111111111111111111111111111, fee_account: 7WdrT3PETDZUDyhUi7Gp8WChGDdqSnAavX6VH1MSpYkS, mint: 1111111ogCyDbaRMvkdsHB3qfdyFYaG1WtRUAf
dh, domain_account: GaZRomHxg8aqZ31PDvpABoeFPyYPWynUQt9G5V4FkKFj, total_sub_created: 0, nft_gated_collection: Some(E5ZnBpH9DYcxRkumKdS4ayJ3Ftb6o3E8wSbXw4N92GWg), max_nft_mint: 2, allow_revoke: false, pric
e_schedule: [Price { length: 1, price: 10000000 }, Price { length: 2, price: 10000000 }] }
Mintrecord Data: MintRecord { tag: MintRecord, count: 0, mint: BM6UwALQkF5yMv85Nv5YiVNVs5wgiRX2qTRPGbCSbvfF }
test test_not_owner_nft_remove ... ok
```

BVSS

AO:A/AC:L/AX:L/C:N/I:L/A:N/D:C/Y:N/R:N/S:U (10.0)

Recommendation

To address this issue, it is recommended to implement a validation step to ensure that the mint associated with the NFT account matches the mint of the **mint_record** account.

Remediation Plan

SOLVED: The **Bonfida team** solved the issue by making a modification to nft_owner_revoke handler

instruction. Now, it validates that the **mint_record** account matches the mint associated to the NFT account provided.

Remediation Hash

https://github.com/Bonfida/sub-registrar/pull/8/commits/5dad244fe04380008aa7147ecd7b98b7c98d8906

7.2 LOSS OF LAMPORTS BY THE OWNER WHEN THE ADMIN REVOKES

// MEDIUM

Description

The `AdminRevoke` instruction allows the registrar authority to revoke any subdomain if the registrar was created with the allow_revoke feature set to true. This instruction requires providing the sub_owner account, among others. However, the **lamports_target_account** provided to the revoke_unchecked function is set to the registrar **authority** instead of the subdomain owner. Consequently, the lamports from the subdomain deletion are sent to the authority's account rather than the subdomain owner's account, rendering the **sub_owner** account redundant.

admin_revoke.rs

```
26
27  #[derive(InstructionsAccount)]
28  pub struct Accounts<'a, T> {
29      #[cons(writable)]
30      /// The registrar account
31      pub registrar: &'a T,
32
33      #[cons(writable)]
34      /// The subdomain account to create
35      pub sub_domain_account: &'a T,
36
37      #[cons(writable)]
38      /// The subrecord account
39      pub sub_record: &'a T,
40
41      /// The current sub domain owner
42      pub sub_owner: &'a T,
43
44      /// The parent domain
45      pub parent_domain: &'a T,
```

```
124 revoke_unchecked::revoke_unchecked(
125         registrar,
126         sub_record,
127         mr,
128         accounts.registrar,
129         accounts.sub_domain_account,
130         accounts.parent_domain,
131         accounts.name_class,
132         accounts.spl_name_service,
```

```
        accounts.sub_record,
        accounts.authority,
        mr_acc,
    )?;
```

Proof of Concept

1. The registrar is created
2. A user register a subdomain
3. The admin revokes the previous subdomain registered
4. Check the registrar authority account lamports

```
registrar Data: Account { lamports: 2345520, data.len: 209, owner: 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs, executable: false, rent_epoch: 18446744073709551615, data: 01ff1f8755e880c5565f9672bcb3c5fc
042868892f4f9d18143d7abcd3020dc6ad962b18cf57ded586629f8fd5c9e7c19ae76397472ce6c0260bd73cf15d1235 }
[*]Bob Register
[2024-05-21T18:48:00.548867000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs invoke [1]
[2024-05-21T18:48:00.550197000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.550209000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.550217000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.550223000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+] Instruction: Register Instruction
[2024-05-21T18:48:00.551743000Z DEBUG solana_runtime::message_processor::stable_log] Program log: collection verified
[2024-05-21T18:48:00.551754000Z DEBUG solana_runtime::message_processor::stable_log] Program log: collection key checke
[2024-05-21T18:48:00.552220000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [2]
[2024-05-21T18:48:00.552250000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.552810000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Get Domain Price..
[2024-05-21T18:48:00.552960000Z DEBUG solana_runtime::message_processor::stable_log] Program log: len: 3
[2024-05-21T18:48:00.553038000Z DEBUG solana_runtime::message_processor::stable_log] Program log: price index: 1
[2024-05-21T18:48:00.553332000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-05-21T18:48:00.553647000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
[2024-05-21T18:48:00.554330000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4645 of 169925 compute units
[2024-05-21T18:48:00.554357000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-05-21T18:48:00.554681000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-05-21T18:48:00.554997000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer
[2024-05-21T18:48:00.555647000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4645 of 162916 compute units
[2024-05-21T18:48:00.555665000Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-05-21T18:48:00.556295000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [2]
[2024-05-21T18:48:00.556708000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.556720000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.556774000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.556782000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Create
[2024-05-21T18:48:00.557536000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [3]
[2024-05-21T18:48:00.557555000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.557794000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [3]
[2024-05-21T18:48:00.557811000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.558138000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [3]
[2024-05-21T18:48:00.558156000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.558300000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 15014 of 154302 compute units
[2024-05-21T18:48:00.558320000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:48:00.561912000Z DEBUG solana_runtime::message_processor::stable_log] Program jCebN34bUfdeUYJT13J1yG16XWQpt5PDx6Mse9GUqhR invoke [2]
[2024-05-21T18:48:00.562855000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.562867000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.562878000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.562884000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: CreateReverse
[2024-05-21T18:48:00.569439000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [3]
[2024-05-21T18:48:00.569924000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.569935000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.569989000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.569996000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Create
[2024-05-21T18:48:00.570673000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [4]
[2024-05-21T18:48:00.570694000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.570927000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [4]
[2024-05-21T18:48:00.570943000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.571258000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [4]
[2024-05-21T18:48:00.571275000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.571421000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 12095 of 69929 compute units
[2024-05-21T18:48:00.571442000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:48:00.572031000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [3]
[2024-05-21T18:48:00.572283000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.572294000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.572340000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.572347000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Update Data
```

```
[2024-05-21T18:48:00.572031000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [3]
[2024-05-21T18:48:00.572283000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.572294000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.572340000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.572347000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Update Data
[2024-05-21T18:48:00.572574000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 2324 of 54286 compute units
[2024-05-21T18:48:00.572591000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:48:00.572888000Z DEBUG solana_runtime::message_processor::stable_log] Program jCebN34bUfdeUYJT13J1yG16XWQpt5PDx6Mse9GUqhR consumed 61694 of 112597 compute units
[2024-05-21T18:48:00.572908000Z DEBUG solana_runtime::message_processor::stable_log] Program jCebN34bUfdeUYJT13J1yG16XWQpt5PDx6Mse9GUqhR success
[2024-05-21T18:48:00.573302000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [2]
[2024-05-21T18:48:00.573321000Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-05-21T18:48:00.573607000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs consumed 153402 of 200000 compute units
[2024-05-21T18:48:00.573633000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs success
Subrecord Data: SubDomainRecord { tag: SubRecord, registrar: 65CfpMvNy9BUtdCGdqUxspJZJEUUP1Ds5N6am8iKr3NL, sub_key: AH1mNr2Vy3817V5Pt9DqaViNDtqNi2xzuDK4UBSEWmbx, mint_record: Some(HZRS5XqPQ2BQ6ogGx7pSxnpy
v81hDw9MqPdxPZcpn4TT) }
authority lamports: 100000000000
Bob lamports: 99994125760
[*]Admin Revoke
[2024-05-21T18:48:00.579685000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs invoke [1]
[2024-05-21T18:48:00.580321000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.580330000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.580339000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.580345000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+] Instruction: Admin revoke instruction
[2024-05-21T18:48:00.583156000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Name Record Header owner del sub domian: apghUhdb5xX9s2Bwn7bXEict1trCGS1RXcbJR2dFcJq
[2024-05-21T18:48:00.583164000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+]Transfer Subdomain...
[2024-05-21T18:48:00.583538000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [2]
[2024-05-21T18:48:00.583882000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.583894000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.583914000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.583921000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Transfer Ownership
[2024-05-21T18:48:00.584183000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 2517 of 175612 compute units
[2024-05-21T18:48:00.584202000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:48:00.586279000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Name Record Header owner del sub domian: 65CfpMvNy9BUtdCGdqUxspJZJEUUP1Ds5N6am8iKr3NL
[2024-05-21T18:48:00.586292000Z DEBUG solana_runtime::message_processor::stable_log] Program log: [+]Delete..
[2024-05-21T18:48:00.586552000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX invoke [2]
[2024-05-21T18:48:00.586812000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Entrypoint
[2024-05-21T18:48:00.586823000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Beginning processing
[2024-05-21T18:48:00.586835000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction unpacked
[2024-05-21T18:48:00.586842000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Delete Name
[2024-05-21T18:48:00.587018000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX consumed 1847 of 158185 compute units
[2024-05-21T18:48:00.587039000Z DEBUG solana_runtime::message_processor::stable_log] Program namesLPneVptA9Z5rqUDD9tMTWEJwofgaYwp8cawRkX success
[2024-05-21T18:48:00.587657000Z DEBUG solana_runtime::message_processor::stable_log] Program log: Name Record Header owner del sub domian: 11111111111111111111111111111111
[2024-05-21T18:48:00.587959000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs consumed 48516 of 200000 compute units
[2024-05-21T18:48:00.587981000Z DEBUG solana_runtime::message_processor::stable_log] Program 2KkyPzjaAYaz2ojQZ9P3xYakLd96B5UH6a2isLaZ4Cgs success
authority lamports: 100003132000
Bob lamports: 99994125760
test test_nft ... ok

test result: ok. 1 passed; 0 failed; 0 ignored; 0 measured; 1 filtered out; finished in 0.18s

     Running tests/state.rs (target/debug/deps/state-44b6c7316f5cd008)

running 0 tests
```

BVSS

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

Recommendation

To address this issue, it is advisable to modify the `revoke_unchecked` function call in the `admin_revoke` instruction handler to provide the subdomain owner's account as the **lamports_target_account**. This change ensures that the lamports from the subdomain deletion are correctly credited to the sub_owner's account.

Remediation Plan

SOLVED: The **Bonfida team** solved the issue by modifying the `admin_revoke` handler instruction in the following commits:

- https://github.com/Bonfida/sub-registrar/pull/8/commits/a98039b06f137e4eaf9993310d0fc2031ab96439
- https://github.com/Bonfida/sub-registrar/commit/5815184c1da07c565f7f2f51e8d93c5faeb50269

Now, the **sub_owner** account is verified to match the subdomain owner's account and it is passed as the **lamports_target_account** in the `revoke_unchecked` function call. This ensures that the subdomain owner receives the lamports from the subdomain deletion.

Remediation Hash

https://github.com/Bonfida/sub-registrar/commit/5815184c1da07c565f7f2f51e8d93c5faeb50269

7.3 NFT GATED COLLECTION CHECK MISSING

// LOW

Description

The `CreateRegistrar` instruction allows a domain owner to create a registrar by providing various values and parameters, including the optional **nft_gated_collection** account. This parameter restricts access to records behind an NFT collection, allowing users who own NFTs associated with that collection to register subdomains. Notably, this parameter cannot be modified once the registration is created.
However, there is currently no validation for this account, allowing to be set even to the zero address. If an invalid account is provided mistakenly, it will prevent subdomain registrations from occurring. Since the parameter cannot be changed after creation, the only solution would be to close the registrar and recreate it with the correct account.

create_registrar.rs

```
30 pub struct Params {
31     pub mint: Pubkey,
32     pub fee_account: Pubkey,
33     pub authority: Pubkey,
34     pub price_schedule: Vec<u8>,
35     pub nft_gated_collection: Option<Pubkey>,
36     pub max_nft_mint: u8,
37     pub allow_revoke: bool,
38 }
```

```
118     // Create Registry account
119     let seeds: &[&[u8]] = &[
120         Registrar::SEEDS,
121         &accounts.domain_name_account.key.to_bytes(),
122         &[nonce],
123     ];
124     let registry = Registrar::new(
125         &params.authority,
126         &params.fee_account,
127         &params.mint,
128         accounts.domain_name_account.key,
129         price_schedule,
130         nonce,
131         params.nft_gated_collection,
132         params.max_nft_mint,
133         params.allow_revoke,
134     );
135     Cpi::create_account(
136         program_id,
```

```
138             accounts.system_program,
139             accounts.fee_payer,
140             accounts.registrar,
141             seeds,
142             registry.borsh_len(),
143         )?;
        registry.save(&mut accounts.registrar.data.borrow_mut());
```

BVSS

AO:S/AC:L/AX:L/R:N/S:C/C:N/A:M/I:M/D:C/Y:C (3.8)

Recommendation

To remediate this issue, it is recommended to integrate a validation mechanism for the **nft_gated_collection** parameter within the `create_registrar` handler instruction. This validation should verify that if the **nft_gated_collection** account is provided, it indeed meets the criteria of being an NFT collection, ensuring it is a valid NFT itself. By implementing this validation, it will prevent the parameter from being set to an invalid or zero address, thereby averting potential denial of service issues that may arise from subdomain registrations being hindered.

Remediation Plan

RISK ACCEPTED: The **Bonfida team** accepted the risk of this finding, noting their intention to allow the creation of collection registrars even if the collection address is known but not yet created. If an invalid collection address is provided, only the process will need to be restarted.

7.4 SUBREVERSE ACCOUNT CHECK MISSING IN ADMIN REGISTER

// LOW

Description

The `AdminRegister` instruction allows the registration authority to register subdomains at no cost. Similar to the `Register` instruction, it requires providing several accounts, including the **sub_reverse_account**, to create a reverse account if it does not already exist for the subdomain being registered. However, in the admin_register instruction handler, the **sub_reverse_account** is not completely validated. Consequently, it is possible to provide a reverse account that has already been created for a different subdomain. Since its data is not empty, no reverse account will be created for the subdomain being registered, leading to potential inconsistencies.

admin_register.rs

```
120     // Check owners
121         check_account_owner(accounts.registrar, program_id)?;
122         check_account_owner(accounts.parent_domain_account,
123 &spl_name_service::ID)?;
124         check_account_owner(accounts.sub_domain_account,
125 &system_program::ID)?;
126         check_account_owner(accounts.sub_reverse_account,
    &system_program::ID).or_else(|_| {
                check_account_owner(accounts.sub_reverse_account,
    &spl_name_service::ID)
            })?;
```

```
175  // Create sub
176     let space: u32 = 0;
177     let lamports = Rent::get()?.minimum_balance(space as usize +
178 NameRecordHeader::LEN);
179     let ix = spl_name_service::instruction::create(
180         spl_name_service::ID,
181 
182 spl_name_service::instruction::NameRegistryInstruction::Create {
183             hashed_name,
184             lamports,
185             space,
186         },
187         *accounts.sub_domain_account.key,
188         *accounts.authority.key,
189         *accounts.authority.key,
190         None,
191         Some(registrar.domain_account),
192 
```

```
        Some(*accounts.registrar.key),
    )?;

    let seeds: &[&[u8]] = &[
        Registrar::SEEDS,
        &registrar.domain_account.to_bytes(),
        &[registrar.nonce],
    ];
    invoke_signed(
        &ix,
        &[
            accounts.spl_name_service.clone(),
            accounts.system_program.clone(),
            accounts.authority.clone(),
            accounts.sub_domain_account.clone(),
            accounts.authority.clone(),
            accounts.parent_domain_account.clone(),
            accounts.registrar.clone(),
        ],
        &[seeds],
    )?;

    // Sub reverse should be passed in the accounts and check if
does not already exist
    if accounts.sub_reverse_account.data_is_empty() {
        msg!("empty: create Reverse");
        let ix = create_reverse(
            sns_registrar::ID,
            create_reverse::Accounts {
                naming_service_program: &spl_name_service::ID,
                root_domain: &ROOT_DOMAIN_ACCOUNT,
                reverse_lookup: accounts.sub_reverse_account.key,
                system_program: &system_program::ID,
                central_state: &sns_registrar::central_state::KEY,
                fee_payer: accounts.authority.key,
                rent_sysvar: accounts.rent_sysvar.key,
                parent_name:
Some(accounts.parent_domain_account.key),
                parent_name_owner: Some(accounts.registrar.key),
            },
            create_reverse::Params {
                name: params.domain,
            },
        );
        invoke_signed(
            &ix,
```

```
            &[
                accounts.sns_registrar_program.clone(),
                accounts.spl_name_service.clone(),
                accounts.rent_sysvar.clone(),
                accounts.spl_name_service.clone(),
                accounts.root_domain.clone(),
                accounts.sub_reverse_account.clone(),
                accounts.system_program.clone(),
                accounts.reverse_lookup_class.clone(),
                accounts.authority.clone(),
                accounts.parent_domain_account.clone(),
                accounts.registrar.clone(),
            ],
            &[seeds],
        )?;
    }

    // Create subrecord account
    let sub_record =
        SubDomainRecord::new(*accounts.registrar.key,
*accounts.sub_domain_account.key);
```

BVSS

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

Recommendation

To address this issue, it is recommended to implement additional validation for the **sub_reverse_account** within the `admin_register` instruction handler before checking if the account already exists. This validation should ensure that the provided **sub_reverse_account** corresponds to the subdomain being registered. This prevents the use of a reverse account already associated with a different subdomain, ensuring the accurate and consistent creation of subdomains and reverse accounts.

Remediation Plan

SOLVED: The **Bonfida team** solved this issue by adding a validation in the `admin_register` instruction handler. Now, the provided **sub_reverse_account** is properly checked to ensure it corresponds to the subdomain being registered.

Remediation Hash

https://github.com/Bonfida/sub-registrar/commit/230dfcb74ee4ad17a18b8e447f8c5313e3cd8f23

7.5 HARDCODED FEE ADDRESS IN REGISTER

// INFORMATIONAL

Description

The `Register` instruction allows users to register a subdomain by paying the corresponding price, which includes the scheduled price and a fee. The scheduled price is transferred to the registrar's fee account, while the fee is transferred to a specified account provided in the call. The instruction handler verifies that this specified account matches the hardcoded Bonfida fee address, **FEE_ACC_OWNER**. Since this address is hardcoded, it cannot be modified without redeploying the program if the account becomes compromised or needs to be changed.

state.rs

```
// Fee account
pub const FEE_ACC_OWNER: Pubkey = pubkey!
("5D2zKog251d6KPCyFyLMt3KroWwXXPWSgTPyhV22K2gR");
```

register.rs

```
pub fn process(program_id: &Pubkey, accounts: &[AccountInfo],
params: Params) -> ProgramResult {
    let accounts = Accounts::parse(accounts, program_id)?;
    let (subrecord_key, subrecord_nonce) =
        SubDomainRecord::find_key(accounts.sub_domain_account.key,
program_id);
    let mut registrar =
Registrar::from_account_info(accounts.registrar, Tag::Registrar)?;

    check_account_key(accounts.fee_account,
&registrar.fee_account)?;
    check_account_key(accounts.parent_domain_account,
&registrar.domain_account)?;
    check_account_key(accounts.sub_record, &subrecord_key)?;
    check_token_account_owner(accounts.bonfida_fee_account,
&FEE_ACC_OWNER)?;
```

BVSS

AO:S/AC:L/AX:L/C:N/I:L/A:M/D:N/Y:N/R:N/S:C (1.4)

Recommendation

To enhance flexibility and security, consider replacing the hardcoded fee address with a configuration account or registry that can be only updated by authorized administrators. Implement validation checks

when updating the fee address to ensure it meets all necessary criteria and prevent unauthorized changes to not been compromised.

Remediation Plan

ACKNOWLEDGED: The **Bonfida team** acknowledged this finding.

7.6 AUTHORITY ADDRESS CHECK MISSING

// INFORMATIONAL

Description

The `CreateRegistrar` instruction enables a domain owner to create a registrar, requiring several parameters, including an authority account designated as the registrar authority. However, this account is not validated allowing to be set even to the zero address. If an invalid account is mistakenly provided, it cannot be modified later because the `EditRegistrar` instruction can only be called by this registrar authority. This also prevents the ability to close the register and call `AdminRevoke` or `AdminRegister`. The same issue applies to the `EditRegistrar` instruction.

create_registrar.rs

```
30  pub struct Params {
31      pub mint: Pubkey,
32      pub fee_account: Pubkey,
33      pub authority: Pubkey,
34      pub price_schedule: Vec<u8>,
35      pub nft_gated_collection: Option<Pubkey>,
36      pub max_nft_mint: u8,
37      pub allow_revoke: bool,
38  }
```

```
117 // Create Registry account
118     let seeds: &[&[u8]] = &[
119         Registrar::SEEDS,
120         &accounts.domain_name_account.key.to_bytes(),
121         &[nonce],
122     ];
123     let registry = Registrar::new(
124         &params.authority,
125         &params.fee_account,
126         &params.mint,
127         accounts.domain_name_account.key,
128         price_schedule,
129         nonce,
130         params.nft_gated_collection,
131         params.max_nft_mint,
132         params.allow_revoke,
133     );
```

edit_registrar.rs

```rust
pub struct Accounts<'a, T> {
    /// The system program account
    pub system_program: &'a T,

    #[cons(writable, signer)]
    /// The fee payer account
    pub authority: &'a T,

    #[cons(writable)]
    /// The registry to edit
    pub registrar: &'a T,
```

```rust

pub fn process(program_id: &Pubkey, accounts: &[AccountInfo],
params: Params) -> ProgramResult {
    let accounts = Accounts::parse(accounts, program_id)?;
    let mut registrar =
Registrar::from_account_info(accounts.registrar, Tag::Registrar)?;

    check_account_key(accounts.authority, &registrar.authority)?;

    if let Some(new_authority) = params.new_authority {
        registrar.authority = new_authority;
    }
```

BVSS

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

Recommendation

To address the issue, it is recommended to implement a validation check for the authority account in the `CreateRegister` and `EditRegister` instructions, to ensure that the provided account meets all necessary criteria and exists within the system.

Remediation Plan

ACKNOWLEDGED: The **Bonfida team** acknowledged this finding.

7.7 POTENTIAL SUBDOMAIN IMPERSONATION

// INFORMATIONAL

Description

The `AdminRevoke` instruction permits the registrar authority to revoke any subdomain that has already been registered. However, a security issue has been identified related to impersonation.
In the current implementation, the registrar authority can revoke a user's subdomain and immediately re-register it. This allows the domain owner to impersonate the user without their knowledge, as there is no expiration period to alert the user of the impending revocation. Consequently, the user is unable to prepare or respond to the loss of their subdomain.
Note: This issue was identified by the `Bonfida team`.

BVSS

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

Recommendation

To address this issue, it is recommended to implement an expiration period following the revocation of a subdomain, during which the subdomain cannot be re-registered. This will give the affected user sufficient time to respond to the revocation and mitigate the risk of impersonation.

Remediation Plan

SOLVED: The **Bonfida team** solved the issue by adding some changes. A revoke_expiry_time field has been added to the registrar. This field is set at the time of registrar creation and represents the mandatory waiting period that must elapse between the revocation of a subdomain by the registrar authority and the re-registration of that subdomain.
In addition, now it is allowed the deletion of the subrecord account corresponding to the revoked subdomain after the mandatory waiting period mentioned above has elapsed in case it has not been re-registered. The remaining lamports of the account will be deposited to the original owner of the revoked subdomain.

Remediation Hash

https://github.com/Bonfida/sub-registrar/pull/9/commits/e14ab1b5226b8b68cd05c242ae1bf7bbe3408267

7.8 TOKEN MINT AND FEE ACCOUNT CHECK MISSING

// INFORMATIONAL

Description

The `CreateRegistrar` and `EditRegistrar` instructions necessitate certain parameters, including the **fee_account** and **mint_account**. This last account receives the payment when registering. However, both accounts values lack validation.

This allows for the possibility of providing an account as **fee_account** that may not necessarily be an SPL token account. Even if it is, it doesn't need to be associated with the provided **mint_account**, and its authority can be arbitrary.

This renders the mint_account redundant, as it's only necessary that the mint of the **fee_account** and **fee_source** match for the payment during registration, which can differ from the registrar's mint.

create_registrar.rs

```
30 pub struct Params {
31     pub mint: Pubkey,
32     pub fee_account: Pubkey,
33     pub authority: Pubkey,
34     pub price_schedule: Vec<u8>,
35     //esto sirva àra crea un registro con un collección NFT
36     pub nft_gated_collection: Option<Pubkey>,
37     pub max_nft_mint: u8,
38     pub allow_revoke: bool,
39 }
```

```
123     let registry = Registrar::new(
124         &params.authority,
125         &params.fee_account,
126         &params.mint,
127         accounts.domain_name_account.key,
128         price_schedule,
129         nonce,
130         params.nft_gated_collection,
131         params.max_nft_mint,
132         params.allow_revoke,
133     );
```

edit_registrar.rs

```
30 pub struct Params {
31     /// The new registry
32     pub new_authority: Option<Pubkey>,
33     pub new_mint: Option<Pubkey>,
```

```
    pub new_fee_account: Option<Pubkey>,
    pub new_price_schedule: Option<Vec<u8>>,
    pub new_max_nft_mint: Option<u8>,
}
```

```
    if let Some(new_mint) = params.new_mint {
        registrar.mint = new_mint;
    }

    if let Some(new_fee_account) = params.new_fee_account {
        registrar.fee_account = new_fee_account;
    }

```

BVSS

AO:S/AC:L/AX:L/C:N/I:L/A:N/D:N/Y:N/R:N/S:U (0.5)

Recommendation

To mitigate this issue, it is recommended to implement validation for both the mint_account and the fee_account parameters in the CreateRegistrar and EditRegistrar instructions. This validation ensures that the provided fee_account is indeed an SPL token account and that its associated mint matches the specified mint_account.

Remediation Plan

ACKNOWLEDGED: The **Bonfida team** 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-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-2024-0019	mio	Tokens for named pipes may be delivered after deregistration

---

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.