Accretion

SECURITY ASSESSMENT

Provided by Accretion Labs Pte Ltd. for Tessera
January 19, 2026
A25TES1

Tessera

AUDITORS

Role	Name
Lead Auditor	Robert Reith (robert@accretion.xyz)
Auditor	Niklas Brymko (niklas@accretion.xyz)
Auditor	David Ratiney (david@accretion.xyz)
Auditor	J4x (contact@accretion.xyz)

CLIENT

Tessera (https://www.tessera.pe/) is building a way to issue tokens for and trade private companies on-chain. They hired Accretion to conduct a security assessment of ther token program and their referral program.

ENGAGEMENT TIMELINE

02 Dec — Project Kickoff
Initial planning and scope definition

02 Dec — Assessment Begins
Security review and testing phase

14 Dec — Review Fixes
Security recommendations are given and implemented

19 Jan — Project Completion
Report delivery and on-chain confirmation

AUDITED CODE

Program 1

ProgramID: HiA4mhg5viZhiPHsJg2rEo2B5L2TNnNkwDi6AzCT9eD4

Repository: https://github.com/tessera-lab/tessera-on-solana

Program 2

ProgramID: TESQvsR4TmYxiroPPQgZpVRoSFG8pru4fsYr67iv6kf

Repository: https://github.com/tessera-lab/tessera-on-solana

ASSESSMENT

The security assessment of Tessera's Token and Referral Solana programs revealed multiple access control and logic issues requiring attention. We identified 30 security findings: 9 medium, 11 low, and 10 informational. No critical or high-severity vulnerabilities were found, and the team addressed the majority of findings, fixing 17 issues across all severity levels.

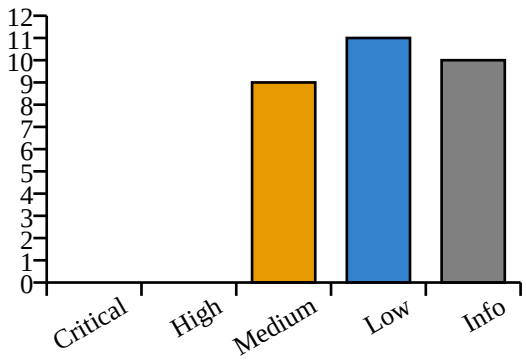
CODE ASSESSMENT

The code implemented a comprehensive referral system with tiered fees and token management functionality using the Anchor framework. However, the access control model had several gaps, particularly around admin privilege management and multisig compatibility. The documentation was noted as redundant and could benefit from consolidation.

KEY FINDINGS

Our top findings centered on access control issues: users could be re-registered breaking the immutable referral chain, old authorities retained admin privileges after removal, admins couldn't manage other admins, and several instructions would fail when transitioning to multisig authorities. Additional issues included the WriteTokenMetadata instruction being non-functional and transfer fees not working without proper FeeManagerConfig initialization.

SEVERITY DISTRIBUTION



ENGAGEMENT SCOPE

The scope of this security assessment was a full review of the following items:

Item 1: Tessera Referral System

Link: https://github.com/tessera-lab/tessera-on-solana
Commit: 7c1eea1a
Program ID: HiA4mhg5viZhiPHsJg2rEo2B5L2TNnNkwDi6AzCT9eD4
Audit Result:
• **Audited Commit:** b2bb96b0741954252754ca64e5d497790367193a
• **Build Hash:** unverified
• **Status:** unverified
• **Comment:** not verified

Item 2: Tessera Token Program

Link: https://github.com/tessera-lab/tessera-on-solana
Commit: 7c1eea1a
Program ID: TESQvsR4TmYxiroPPQgZpVRoSFG8pru4fsYr67iv6kf
Audit Result:
• **Audited Commit:** b2bb96b0741954252754ca64e5d497790367193a
• **Build Hash:** unverified
• **Status:** unverified
• **Comment:** not verified

ISSUES SUMMARY

ID	TITLE	SEVERITY	STATUS
ACC-M1	`WriteTokenMetadata` IX will never work	MEDIUM	FIXED
ACC-M2	Transfer fees won't work without `FeeManagerConfig`	MEDIUM	FIXED
ACC-M3	`withdrawal_threshold` can be bypassed by batching withdrawals	MEDIUM	WONTFIX
ACC-M4	`authority` won't be able to call `AdminCreateReferralCode` once it's a multisig	MEDIUM	FIXED
ACC-M5	Old authority maintains admin privileges after being removed	MEDIUM	FIXED
ACC-M6	Admin can not add/remove other admins	MEDIUM	FIXED
ACC-M7	`authority` lacks privileges in edge case	MEDIUM	FIXED
ACC-M8	User can be re-registered breaking immutable referral chain	MEDIUM	FIXED
ACC-M9	Missing Referrer Registration Check	MEDIUM	FIXED
ACC-L1	Token creation won't work for multisigs	LOW	FIXED
ACC-L2	Transfer fee config can be changed without using the `UpdateTransferFeeConfig` IX	LOW	WONTFIX
ACC-L3	Some referral codes will revert due to not being serialisable	LOW	FIXED
ACC-L4	Missing check for `Tier1Fee >= Tier2Fee >= Tier3Fee`	LOW	FIXED
ACC-L5	`referral_code_owner` will be registered twice in some cases	LOW	WONTFIX
ACC-L6	Rent paid by user goes to admin in case of admin close	LOW	WONTFIX
ACC-L7	Incorrect Closure of Accounts	LOW	WONTFIX
ACC-L8	Program Logs May Be Truncated	LOW	WONTFIX
ACC-L9	TesseraToken ProgramConfig frontrun	LOW	FIXED
ACC-L10	Fee config snapshots current fee tiers.	LOW	WONTFIX
ACC-L11	Frontruning of Referral Program Initialization.	LOW	FIXED
ACC-I1	Incorrect check for set metadata	INFO	FIXED
ACC-I2	Both documentations are bloated and include the same content 2-4x	INFO	WONTFIX
ACC-I3	Incorrect documentation of `update_fee_manager`	INFO	WONTFIX
ACC-I4	Incorrect decimals enforced on mint	INFO	FIXED

ACC-I5	Mint creation can be DOSd	INFO	WONTFIX
ACC-I6	Oversized `SenderFeeConfig` leads to unnecessary rent cost	INFO	WONTFIX
ACC-I7	Admin may close a referral code and user may reinitialize it	INFO	FIXED
ACC-I8	Referrer tier list can be overwritten and contains double record	INFO	FIXED
ACC-I9	Modification after Validation in Create Referral Code	INFO	WONTFIX
ACC-I10	FeeManager ATA is not owner validated	INFO	WONTFIX

DETAILED ISSUES

ACC-M1 `WriteTokenMetadata` IX will never work

MEDIUM FIXED

Description

The `write_token_metadata` function will allow users to initialize the metadata of their token into their mint. The function will call the `initialize_token_metadata` IX on behalf of the user.

```
// Write metadata via CPI
let metadata_ix = initialize_token_metadata(
    &ctx.accounts.token_2022_program.key(),
    &ctx.accounts.mint.key(),
    &ctx.accounts.authority.key(),
    &ctx.accounts.mint.key(),
    &ctx.accounts.authority.key(),
    name.clone(),
    symbol.clone(),
    uri.clone(),
);

invoke(
    &metadata_ix,
    &[
        ctx.accounts.mint.to_account_info(),
        ctx.accounts.authority.to_account_info(),
    ],
)?;
```

This instruction will automatically reallocate the account to the needed size and write the metadata into it.

```
    // allocate a TLV entry for the space and write it in, assumes that there's
    // enough SOL for the new rent-exemption
    alloc_and_serialize_variable_len_extension::<PodMint, _>(
        metadata_info,
        &token_metadata,
        false,
    )?;
```

This will not automatically update the rent to the required amount. The tessera token program tries to calculate this increase and add the additional lamports needed.

```
// Calculate metadata space and fund if needed
// Each string has a 4-byte length prefix, plus struct overhead
let metadata_space = STRING_LENGTH_SIZE
    .checked_add(name.len())
    .and_then(|s| s.checked_add(STRING_LENGTH_SIZE))
    .and_then(|s| s.checked_add(symbol.len()))
    .and_then(|s| s.checked_add(STRING_LENGTH_SIZE))
    .and_then(|s| s.checked_add(uri.len()))
    .and_then(|s| s.checked_add(METADATA_OVERHEAD))
    .ok_or(ErrorCode::InvalidMintSpace)?;

let current_size = ctx.accounts.mint.data_len();
let new_size = current_size
    .checked_add(metadata_space)
    .ok_or(ErrorCode::InvalidMintSpace)?;
```

However, this calculation is incorrect. The overhead that is added is of the following format:

| Field | Size (bytes) | Description | |---|---|---| | Extension Header | 4 | 2 bytes (Type) + 2 bytes (Length) | | Update Authority | 32 | OptionalNonZeroPubkey (exactly 32 bytes) | |

Mint | 32 | Pubkey (exactly 32 bytes) | | Additional Metadata Count | 4 | Vec length prefix (u32) |

As a result, the IX will transfer an insufficient amount of rent to the mint, causing it to fail as the account was resized but does not have enough rent afterwards.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/metadata.rs#L49-L63

Relevant Code

Mitigation Suggestion

We recommend accounting for the correct overhead by changing `METADATA_OVERHEAD` to 72.

```
// Correct calculation:
// 4 (Ext Header) + 32 (UpdateAuth) + 32 (Mint) + 4 (String Len Name) + Name + 4 (String Len Symbol) + Symbol
 + 4 (String Len URI) + URI + 4 (Vec Len)
let metadata_space = 4 + 32 + 32 + 4 + name.len() + 4 + symbol.len() + 4 + uri.len() + 4;
```

Remediation

Fixed in commit `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by adding changing `METADATA_OVERHEAD` to 72.

ACC-M2 Transfer fees won't work without `FeeManagerConfig`

MEDIUM FIXED

Description

Throughout the documentation, it is stated multiple times that the `FeeManagerConfig` is an optional account, and that if it isn't initialised, fee withdrawals will still work.

```
### FeeManagerConfig (Optional Program-Level PDA)

**Purpose**:
- Optional configuration for automated fee collection
- Defines who can withdraw fees
- Sets minimum threshold for withdrawals

**Note**: If not created, fees can still be withdrawn using native Token-2022 withdraw authority.
```

However, this is untrue. On initialization, the `withdraw_withheld_authority` is set to a program-derived PDA. So this PDA is the only way to withdraw fees.

```
// Derive fee_authority PDA
let (fee_authority_pda, _fee_authority_bump) = Pubkey::find_program_address(
    &[FEE_AUTHORITY_SEED, ctx.accounts.mint.key().as_ref()],
    &crate::ID,
);

// Initialize transfer fee extension
let transfer_fee_config_ix = transfer_fee_instruction::initialize_transfer_fee_config(
    &ctx.accounts.token_2022_program.key(),
    &ctx.accounts.mint.key(),
    Some(&ctx.accounts.authority.key()),
    Some(&fee_authority_pda),
    transfer_fee_basis_points,
    maximum_fee,
)?;
```

The only way to get this PDA to sign is to call the `WithdrawToFeeManager` IX, which requires a initialised `FeeManagerConfig` based on its context.

```
#[derive(Accounts)]
pub struct WithdrawToFeeManager<'info> {
    #[account(
        seeds = [b"fee_manager_config"],
        bump = fee_manager_config.bump
    )]
    pub fee_manager_config: Account<'info, FeeManagerConfig>,
```

As a result, once the `FeeManagerConfig` has no withdrawals, it will no longer work, so it's not optional.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/contexts/fee_management.rs#L35-L41

Relevant Code

Mitigation Suggestion

We recommend adding a way to allow the PDA to sign on behalf of the authority without requiring the `FeeManagerConfig` to be initialized.

Remediation

Fixed in `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by updating the documentation.

ACC-M3 `withdrawal_threshold` can be bypassed by batching withdrawals

MEDIUM WONTFIX

Description

The token program implements a safeguard that prevents any caller from transferring more than the fee threshold to the fee manager.

```
/// Minimum amount of fees (in tokens) that can be withdrawn without authority signature
/// Withdrawals above this threshold require authority to sign
pub withdrawal_threshold: u64,
```

This value is then enforced as a constraint compared to the `transfer_fee_config.withheld_amount`.

```
// Verify authority if above threshold
let config = &ctx.accounts.fee_manager_config;
if withheld_amount > config.withdrawal_threshold {
    let authority_opt = &ctx.accounts.authority;
    require!(authority_opt.is_some(), ErrorCode::UnauthorizedWithdrawal);

    let authority = authority_opt.as_ref().unwrap();
    require!(
        authority.key() == config.authority,
        ErrorCode::InvalidAuthority
    );
}
```

This value is the current total of fees moved to the mint. The fees are collected to the mint because the caller can pass an arbitrary list of remaining accounts to the IX, and for each account, `harvest_withheld_tokens_to_mint` is called, and its withheld tokens are moved to the mint.

This leads to an issue: the caller (who can be anyone) is not required to collect all currently withheld tokens at once. The attacker can pass only so many remaining accounts so that the sum of their withheld tokens stays below the threshold. Then he can chain multiple of these IX to move significantly more than the threshold without needing an authority signature. This will allow the attacker to bypass this safeguard completely due to its atomicity.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/withdraw_fees.rs#L69-L80

Relevant Code

Mitigation Suggestion

We recommend implementing a non-atomic system. One solution would be to set a maximum that can be moved without an authority signature, and then deduct every move from that. This value should then gradually be refilled over time.

Remediation

Marked as won't fix due to the overhead a mitigation would bring.

ACC-M4 `authority` won't be able to call `AdminCreateReferralCode` once it's a multisig

MEDIUM FIXED

Description

The documentation describes that, post initialization, the `authority` role should be transferred to a multisig.

```
1. Primary Authority (Deployer)
Description: Root authority with full control over the system

Storage: ReferralConfig.authority

Capabilities:

■ Initialize referral system (one-time)
■ Update tier percentages
■ Update default fee recipient
■ Transfer authority to another address (e.g., multisig) <---
■ Add/remove admins
■ Cannot be removed from admin list
■ Has all admin privileges
Security:

Should be transferred to a multisig for production <---
Only authority that can transfer itself
Protected from removal in admin list operations
```

The `authority` will also get admin privileges and should be able to call all admin-restricted functions. However, it cannot call the `AdminCreateReferralCode` instruction if it is a PDA-based multisig due to the following constraint.

```
#[account(
    init,
    payer = admin,
    space = DISCRIMINATOR_SIZE + ReferralCode::LEN,
    seeds = [b"referral_code", normalize_referral_code(&code).as_bytes()],
    bump
)]
pub referral_code: Account<'info, ReferralCode>,
```

The `payer` constraint requires the `admin` to sign a transfer instruction, which a PDA can't do. As a result, any call to this by the multisig authority will fail.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/contexts/referral_code.rs#L43

Relevant Code

Mitigation Suggestion

We recommend implementing the same structure as in `AdminRegisterWithReferralCode`, where a separate payer signs and is used as the payer for account creation.

Remediation

Fixed in commit `b2bb96b0741954252754ca64e5d497790367193a`.

ACC-M5 Old authority maintains admin privileges after being removed

MEDIUM FIXED

Description

On creation, the authority of the referral config is pushed into the admin's list.

```
// Initialize admin list with authority as the only admin
let admin_list = &mut ctx.accounts.admin_list;
admin_list.admins = vec![ctx.accounts.authority.key()];
admin_list.bump = ctx.bumps.admin_list;
```

However, when the authority gets transferred, the old authority is not removed from the admin list, letting it retain its admin privileges.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/instructions/initialization.rs#L147-L164

Relevant Code

```
pub fn update_authority(ctx: Context<UpdateReferralConfig>, new_authority: Pubkey) -> Result<()> {
    // Validate new authority is not the default pubkey
    require!(
        new_authority != Pubkey::default(),
        crate::ErrorCode::InvalidDefaultRecipient // Reuse this error or create a new one
    );

    let config = &mut ctx.accounts.referral_config;
    let old_authority = config.authority;

    // Update authority
    config.authority = new_authority;

    emit!(AuthorityTransferred {
        old_authority,
        new_authority,
        timestamp: Clock::get()?.unix_timestamp,
    });
```

Mitigation Suggestion

We recommend removing the old authority from the admins list in case of authority transfer

Remediation

Fixed in commit `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by removing old authority in authority transfer.

ACC-M6 Admin can not add/remove other admins

MEDIUM FIXED

Description

The documentation states that the admin should be able to add/remove other admins.

```
Description: Delegated authorities for operational management

Storage: AdminList.admins (up to 10 admins)

Capabilities:
■ Add/remove other admins (not primary authority)
```

However this is actually not possible as the admins can only be added or removed by the `authority` due to the following check.

```
#[account(
    seeds = [b"referral_config"],
    bump = referral_config.bump,
    has_one = authority
)]
pub referral_config: Account<'info, ReferralConfig>,

#[account(mut)]
pub authority: Signer<'info>,
```

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/contexts/admin.rs#L16-L24

Relevant Code

Mitigation Suggestion

We recommend adapting the checks to allow admins to also call the 2 functions.

Remediation

Fixed in `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by updating the documentation.

ACC-M7 `authority` lacks privileges in edge case

MEDIUM FIXED

Description

The documentation states the following about the `authority` role:

```
Capabilities:
- // Irrelevant
- ■ Has all admin privileges
```

This is also correctly ensured for the authority at the start, as on initialization it is pushed into the admin list.

```
// Initialize admin list with authority as the only admin
let admin_list = &mut ctx.accounts.admin_list;
admin_list.admins = vec![ctx.accounts.authority.key()];
admin_list.bump = ctx.bumps.admin_list;
```

However the problem occurs if the authority gets transferred using `update_authority()`.

```
pub fn update_authority(ctx: Context<UpdateReferralConfig>, new_authority: Pubkey) -> Result<()> {
    // Validate new authority is not the default pubkey
    require!(
        new_authority != Pubkey::default(),
        crate::ErrorCode::InvalidDefaultRecipient // Reuse this error or create a new one
    );

    let config = &mut ctx.accounts.referral_config;
    let old_authority = config.authority;

    // Update authority
    config.authority = new_authority;
```

In that case, the config authority is changed, which now allows removing the old one from the admin list, but does not automatically push the new one into the admin list. As a result, the `authority` will not have admin privileges until it is manually pushed into the admins list.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/instructions/initialization.rs#L147-L164

Relevant Code

Mitigation Suggestion

We recommend pushing the new authority into the admins list on transfer of authority.

Remediation

Fixed in commit `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by updating the admins list on authority transfer

ACC-M8 User can be re-registered breaking immutable referral chain

MEDIUM FIXED

Description

The documentation states the following about the `UserRegistration` account.

```
Created: When user registers with code
Never Updated: Referral chain is immutable
Never Closed: Permanent record (prevents re-registration)
```

However, the immutable referral chain is not secured in code. Actually, users can register again after registering and thus update their Tier1-3 referrers and referral code at will. This is the case because the context sets the `init_if_needed` constraint for the account. As a result, if it has already been created by the prior registration, the call will still pass and allow for overwriting of the referrers.

```
/// User registration (mint-agnostic)
#[account(
    init_if_needed,
    payer = user,
    space = DISCRIMINATOR_SIZE + UserRegistration::LEN,
    seeds = [b"user_registration", user.key().as_ref()],
    bump
)]
pub user_registration: Account<'info, UserRegistration>,
```

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/contexts/registration.rs#L15-L23

Relevant Code

Mitigation Suggestion

We recommend using `init` instead of `init_if_needed` for the UserRegistration. This will prevent re-registrations.

Remediation

Fixed in `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by updating the documentation.

ACC-M9 Missing Referrer Registration Check

MEDIUM FIXED

Description

We found that on Registration, the optional account `ReferrerRegistration` is not checked to be the correct registration of the referrer. This can lead to miscounting of referrals.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/contexts/registration.rs#L31-L33

Relevant Code

```
/// registration.rs L31-L33
    /// The direct referrer's registration (if they exist in the system)
    /// CHECK: This account may not exist if the direct referrer is not registered
    pub referrer_registration: Option<Account<'info, UserRegistration>>,
```

Mitigation Suggestion

Check that, if the account is provided, the `referrer_registration` matches the actual referrer.

Remediation

Fixed in commit `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by checking the referrer registration matches the owner.

ACC-L1 Token creation won't work for multisigs

LOW FIXED

Description

The `InitializeMintWithTransferFee` and `WriteTokenMetadata` IX will both require the signer to execute a transfer instructions. `InitializeMintWithTransferFee` will do this as part of `create_account` and `WriteTokenMetadata` directly calls the IX on behalf of the user to transfer the missing rent.

```
anchor_lang::system_program::transfer(
    CpiContext::new(
        ctx.accounts.system_program.to_account_info(),
        anchor_lang::system_program::Transfer {
            from: ctx.accounts.authority.to_account_info(),
            to: ctx.accounts.mint.to_account_info(),
        },
    ),
    additional_rent,
)?;
```

However this will exclude users using multisigs from using the program as PDA based multisigs can't call the transfer instruction.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/metadata.rs#L80-L90

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/initialize_mint.rs#L47-L62

Mitigation Suggestion

We recommend adapting the program to allow a seperate payer account which can be any user provided EOA to call the transfer while the original user is still used for all other tasks.

Remediation

Fixed in `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by adding payer accounts to the affected instructions.

ACC-L2 Transfer fee config can be changed without using the `UpdateTransferFeeConfig` IX

LOW WONTFIX

Description

The `UpdateTransferFeeConfig` IX is intended to be used to modify the transfer fee configuration. It performs sanity checks on the fee's BPS value and emits an event at the end.

```
// Validate basis points
validate_basis_points(new_transfer_fee_basis_points)?;
```

```
emit!(TransferFeeConfigUpdated {
    mint: ctx.accounts.mint.key(),
    authority: ctx.accounts.authority.key(),
    transfer_fee_basis_points: new_transfer_fee_basis_points,
    maximum_fee: new_maximum_fee,
    timestamp: Clock::get()?.unix_timestamp,
});
```

The function will effectively call the `set_transfer_fee` function on behalf of the `authority`.

```
// Update via CPI
let update_fee_config_ix =
    anchor_spl::token_2022::spl_token_2022::extension::transfer_fee::instruction::set_transfer_fee(
        &ctx.accounts.token_2022_program.key(),
        &ctx.accounts.mint.key(),
        &ctx.accounts.authority.key(),
        &[],
        new_transfer_fee_basis_points,
        new_maximum_fee,
    )?;

anchor_lang::solana_program::program::invoke(
    &update_fee_config_ix,
    &[
        ctx.accounts.mint.to_account_info(),
        ctx.accounts.authority.to_account_info(),
    ],
)?;
```

As the user-controlled `authority` is set as the `transfer_fee_config_authority` directly on the mint, it can also call this function directly without using the provided one. This will bypass the sanity checks and the event emission.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/update_fee_config.rs#L12

Relevant Code

Mitigation Suggestion

We recommend using a PDA as the `transfer_fee_config_authority` to only allow calling through this endpoint.

Remediation

Issue was marked as won't fix.

ACC-L3 Some referral codes will revert due to not being serialisable

LOW FIXED

Description

The `CreateReferralCode` IX is used to create a referral code. The user will pass a raw string of the code. Then the program will do three steps.

1. Call `validate_referral_code()` to ensure the code is valid and correctly sized 2. `normalize_referral_code` to make it uppercase 3. Call `initialize_referral_code` to initialize the account

`validate_referral_code()` will ensure the following:

```
pub fn validate_referral_code(code: &str) -> Result<()> {
    require!(
        code.len() >= crate::MIN_CODE_LENGTH && code.len() <= crate::MAX_CODE_LENGTH,
        ErrorCode::InvalidCodeFormat
    );
    require!(
        code.chars().all(|c| c.is_alphanumeric()),
        ErrorCode::InvalidCodeFormat
    );
    Ok(())
}
```

This will check that the (not yet uppercased string) is below the max length and is alphanumeric. The alphanumeric check ensures that each character is either a letter of any supported alphabet or a number. This is not just the ASCII alphabet, which is essential. After this, `normalize_referral_code` will be called to transform the string to uppercase. This will call the String's `to_uppercase` function.

```
/// Normalizes a referral code to uppercase for case-insensitive matching
pub fn normalize_referral_code(code: &str) -> String {
    code.to_uppercase()
}
```

This function has its quirks, especially with non-ASCII characters. As one can see from the comment above the function, some letters, for example, the German ß, get turned into two letters when uppercased.

```
/// One character can become multiple:
/// ```
/// let s = "tschüß";
///
/// assert_eq!("TSCHÜSS", s.to_uppercase());
/// ```
```

This leads to a big problem: the size is checked against the non-uppercase string, and these letter types are allowed by `is_alphanumeric()`. For example, the referral code `12345678910ß` would pass because it is exactly 12 characters long. However, once uppercased, it will be turned into `12345678910SS`, which is 13 letters long. When the code is then saved to the account, which is only allocated 12 chars, the write will fail, leading to a panic.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/instructions/referral_code.rs#L11

Relevant Code

Mitigation Suggestion

We recommend two mitigations depending on the intended behavior:

1. If only ASCII chars are intended, `is_alphanumeric()` can be replaced with `is_ascii_alphanumeric()` 2. If special characters are intended to be supported, we recommend verifying the correct length post the call to `to_uppercase` to ensure correct length afterwards.

Remediation

Fixed in commit `c77114e57b4edd6bc0a0fccde216b2a6d3487d8b`.

ACC-L4 Missing check for `Tier1Fee >= Tier2Fee >= Tier3Fee`

LOW FIXED

Description

The `initialize_system()` and `update_config()` instructions allow the admin to set the three fee tiers. According to the documentation, the intention is that the fee BPS distributed to the holder decreases with each tier. If this weren't the case, users could build complex structures to avoid using their own referral code and instead use another user's referral code that they referred, which would go against the intended behavior of the system.

However, in the current implementation of the two functions, neither checks for this, so it allows setting, for example, `Tier2Fee > Tier1Fee`. This is because `validate_tier_percentages` only checks whether each percentage, as well as their sum, is <= 100%.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/9c85bf539095acc77d2fc9703b9adb401bfd164b/programs/referral-system/src/helpers.rs#L58-L75

Relevant Code

```
/// Validates tier percentages don't exceed 100% combined
pub fn validate_tier_percentages(tier1: u16, tier2: u16, tier3: u16) -> Result<()> {
    validate_percentage(tier1)?;
    validate_percentage(tier2)?;
    validate_percentage(tier3)?;

    // Use checked arithmetic to prevent overflow before comparison
    let total = tier1
        .checked_add(tier2)
        .and_then(|sum| sum.checked_add(tier3))
        .ok_or(ErrorCode::Overflow)?;

    require!(
        total <= crate::BASIS_POINTS_MAX,
        ErrorCode::InvalidPercentage
    );
    Ok(())
}
```

Mitigation Suggestion

We recommend adding a check that ensures that `Tier1Fee >= Tier2Fee >= Tier3Fee`

Remediation

Fixed in `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by adding the suggested check.

ACC-L5 `referral_code_owner` will be registered twice in some cases

LOW WONTFIX

Description

When building the referral chain for the new users registration, the process will first set the `referral_code_owner` as the `tier1_referrer`.

```
user_registration.tier1_referrer = referral_code_owner;
```

Then in 2/3 of the following cases it will also set it as the `tier2_referrer`.

```
    } else {
        user_registration.tier2_referrer = referral_code_owner;
        user_registration.tier3_referrer = Pubkey::default();
    }
} else {
    user_registration.tier2_referrer = referral_code_owner;
    user_registration.tier3_referrer = Pubkey::default();
}
```

This is redundant as if `tier1_referrer == tier2_referrer` the `tier2_referrer` will be ignored on the fee distribution.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/helpers.rs#L110-L116

Relevant Code

```
pub fn build_referral_chain(
    user_registration: &mut UserRegistration,
    referral_code_owner: Pubkey,
    referrer_registration: Option<&UserRegistration>,
    user_pubkey: Pubkey,
) -> Result<()> {
    user_registration.tier1_referrer = referral_code_owner;

    if let Some(referrer) = referrer_registration {
        if referrer.is_active {
            // Check for circular references: user cannot appear in their own chain
            require!(
                referrer.tier1_referrer != user_pubkey
                    && referrer.tier2_referrer != user_pubkey
                    && referrer.tier3_referrer != user_pubkey,
                ErrorCode::CannotUseSelfReferralCode
            );

            // Shift the chain: new user becomes tier1, others move down
            user_registration.tier2_referrer = referrer.tier1_referrer;
            user_registration.tier3_referrer = referrer.tier2_referrer;
        } else {
            user_registration.tier2_referrer = referral_code_owner;
            user_registration.tier3_referrer = Pubkey::default();
        }
    } else {
        user_registration.tier2_referrer = referral_code_owner;
        user_registration.tier3_referrer = Pubkey::default();
    }
```

```
    Ok(())
}
```

Mitigation Suggestion

We recommend setting `tier2_referrer = Pubkey::default()` in the cases where it currently is set to the `referral_code_owner`

Remediation

Issue was marked as won't fix.

ACC-L6 Rent paid by user goes to admin in case of admin close

LOW WONTFIX

Description

The `referral_code` part of the codebase allows users and the admin to create referral code accounts. In case of a creation through `create()` the user covers the rent for the account (0.0013224 SOL -> 20c at the current SOL price).

```
pub struct CreateReferralCode<'info> {
    #[account(
        init,
        payer = owner,
        space = DISCRIMINATOR_SIZE + ReferralCode::LEN,
        seeds = [b"referral_code", normalize_referral_code(&code).as_bytes()],
        bump
    )]
    pub referral_code: Account<'info, ReferralCode>,
```

However, when the admin closes the account, which is the only way the account can be closed, the rent is refunded to the admin instead of the actual paying user.

```
pub struct AdminCloseReferralCode<'info> {
    #[account(
        mut,
        seeds = [b"referral_code", referral_code.code.as_bytes()],
        bump = referral_code.bump,
        close = admin
    )]
    pub referral_code: Account<'info, ReferralCode>,
```

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/contexts/referral_code.rs#L60-L67

Relevant Code

Mitigation Suggestion

We recommend saving a `payer` variable in the `ReferralCode` struct and refunding the rent on closure to that account. However, it is essential to ensure that this account can't be set to anything other than a normal address, because if it were, for example, a program account, closure could be blocked that way.

Remediation

Issue was marked as won't fix.

ACC-L7 Incorrect Closure of Accounts

LOW WONTFIX

Description

We found that both programs incorrectly close accounts by removing all lamports, and then in the case of closing Admin List, zeroing out the account data. In the case of closing the fee manager, only lamports are removed. This means that someone may transfer tokens into the account-to-be-closed in order to revive it. In the case of the fee manager for example, the program can be made to emit an event that the fee manager has been closed, even though it may still exist afterward.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/instructions/admin.rs#L14-L23 https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/update_fee_manager.rs#L47-L53

Relevant Code

```
/// admin.rs L14-L23
    // Transfer all lamports from admin_list to authority
    let dest_starting_lamports = authority_info.lamports();
    **authority_info.lamports.borrow_mut() = dest_starting_lamports
        .checked_add(admin_list_info.lamports())
        .unwrap();
    **admin_list_info.lamports.borrow_mut() = 0;

    // Zero out the account data
    let mut data = admin_list_info.try_borrow_mut_data()?;
    data.fill(0);

/// update_fee_manager.rs L47-L53
            **ctx.accounts.authority.lamports.borrow_mut() = new_dest_lamports;
            **ctx
                .accounts
                .fee_manager_config
                .to_account_info()
                .lamports
                .borrow_mut() = 0;
```

Mitigation Suggestion

To properly close accounts, use the anchor `close` constraint.

Remediation

Issue was marked as won't fix.

ACC-L8 Program Logs May Be Truncated

LOW WONTFIX

Description

We found that the program uses the anchor `emit!` macro to log program events. However, solana logs may be truncated if the total log size in the same transaction exceeds 10kb. This means that logs may be only printed partially, or not at all, which can lead to issues in monitoring, debugging, indexing, and compliance.

Location

Every instruction follows this pattern:

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/instructions/initialization.rs#L133-L137

Relevant Code

```
/// initialization.rs L133-L137
    emit!(DefaultFeeRecipientUpdated {
        old_recipient,
        new_recipient: new_default_fee_recipient,
        updated_by: ctx.accounts.authority.key(),
    });
```

Mitigation Suggestion

To mitigate the issue, use Anchor's emit_cpi! macro instead of the emit! macro for crucial event logs. This will ensure that logs will be passed as call data, which are not subject to truncation.

Remediation

Issue was marked as won't fix.

ACC-L9 TesseraToken ProgramConfig frontrun

LOW FIXED

Description

We found similar to #3 in the Referral Program that in the Tessera Token Program, during the program initialization, an attacker can frontrun the global initialization, setting themselves as the admin with control over the fee distributer and configuration updates.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/initialize_program_config.rs#L7-L38

Relevant Code

```
/// initialize_program_config.rs L7-L38
pub fn initialize_program_config(
    ctx: Context<InitializeProgramConfig>,
    fee_manager: Pubkey,
    withdrawal_threshold: u64,
) -> Result<()> {
    // Validate fee manager is not default pubkey
    validate_fee_manager(fee_manager)?;

    // Note: withdrawal_threshold = 0 is allowed and means ALL withdrawals require authority signature
    // This is the most secure configuration but requires authority for every withdrawal

    let fee_manager_config = &mut ctx.accounts.fee_manager_config;
    fee_manager_config.fee_manager = fee_manager;
    fee_manager_config.authority = ctx.accounts.authority.key();
    fee_manager_config.withdrawal_threshold = withdrawal_threshold;
    fee_manager_config.bump = ctx.bumps.fee_manager_config;

    emit!(ProgramConfigInitialized {
        fee_manager,
        authority: ctx.accounts.authority.key(),
        withdrawal_threshold,
        timestamp: Clock::get()?.unix_timestamp,
    });

    msg!(
        "Program config initialized: fee_manager={}, threshold={}",
        fee_manager,
        withdrawal_threshold
    );

    Ok(())
}
```

Mitigation Suggestion

Option 1: Restrict this instruction to be only callable with the signature of the upgrade_authority and the initial admin.
Option 2: Create a hardcoded initial admin.

Remediation

Remediated in 73d59a8b643c81a32ad4bc09160a6bae3a9eea74

ACC-L10 Fee config snapshots current fee tiers.

LOW WONTFIX

Description

We found that `SenderFeeConfig` snapshots the fee configuration. If the tier configuration changes, old referrals will not reflect the current configuration.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/state/accounts.rs#L3-L16

Relevant Code

```
/// accounts.rs L3-L16
/// Fee recipient information for distributing fees
#[derive(AnchorSerialize, AnchorDeserialize, Clone, Debug, PartialEq)]
pub struct FeeRecipientInfo {
    pub recipient: Pubkey,
    pub split_percentage: u16, // Basis points (0-10000)
}

/// Sender-specific fee configuration for multi-tier referral distribution
#[account]
pub struct SenderFeeConfig {
    pub sender: Pubkey,
    pub fee_recipients: Vec<FeeRecipientInfo>,
    pub bump: u8,
}
```

Mitigation Suggestion

Instead of storing the `split_percentage` in `FeeRecipientInfo`, use the index in the `fee_recipients` field of the `SenderFeeConfig` to dictate the tier level and then use the `ReferralConfig` to look up fee tier ratios.

Remediation

Issue was marked as won't fix.

ACC-L11 Frontruning of Referral Program Initialization.

LOW FIXED

Description

We found that the referral system can be frontrun during protocol initialization. An attacker can monitor the chain for program deployment from the initialization with their own transaction, setting themselves as the admin with control over tier split percentages, fee recipient and admin list.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/instructions/initialization.rs#L7-L61

Relevant Code

```
/// initialization.rs L7-L61
pub fn initialize_system(
    ctx: Context<InitializeReferralSystem>,
    default_fee_recipient: Pubkey,
    tier1_split_percentage: u16,
    tier2_split_percentage: u16,
    tier3_split_percentage: u16,
) -> Result<()> {
    // Validate default fee recipient is not the default pubkey
    require!(
        default_fee_recipient != Pubkey::default(),
        crate::ErrorCode::InvalidDefaultRecipient
    );

    validate_tier_percentages(
        tier1_split_percentage,
        tier2_split_percentage,
        tier3_split_percentage,
    )?;

    let referral_config = &mut ctx.accounts.referral_config;
    referral_config.authority = ctx.accounts.authority.key();
    referral_config.default_fee_recipient = default_fee_recipient;
    referral_config.tier1_split_percentage = tier1_split_percentage;
    referral_config.tier2_split_percentage = tier2_split_percentage;
    referral_config.tier3_split_percentage = tier3_split_percentage;
    referral_config.bump = ctx.bumps.referral_config;

    // Initialize admin list with authority as the only admin
    let admin_list = &mut ctx.accounts.admin_list;
    admin_list.admins = vec![ctx.accounts.authority.key()];
    admin_list.bump = ctx.bumps.admin_list;

    msg!(
        "Referral system initialized with authority: {}",
        ctx.accounts.authority.key()
    );
    msg!("Default fee recipient: {}", default_fee_recipient);
    msg!(
        "Tier 1 split: {}",
        basis_points_to_percentage(tier1_split_percentage)
    );
    msg!(
        "Tier 2 split: {}",
        basis_points_to_percentage(tier2_split_percentage)
    );
    msg!(
        "Tier 3 split: {}",
        basis_points_to_percentage(tier3_split_percentage)
```

```
    );
    msg!(
        "Admin list initialized with {} as the first admin",
        ctx.accounts.authority.key()
    );
    Ok(())
}
```

Mitigation Suggestion

Option 1: Restrict this instruction to be only callable with the signature of the upgrade_authority and the initial admin.
Option 2: Create a hardcoded initial admin.

Remediation

Remediated in 73d59a8b643c81a32ad4bc09160a6bae3a9eea74

ACC-I1 Incorrect check for set metadata

INFO FIXED

Description

The `WriteTokenMetadata` instruction includes a check that currently doesn't do anything.

```
// Check if metadata pointer extension exists and points to data
if let Ok(metadata_pointer) = mint.get_extension::<MetadataPointer>() {
    let metadata_address: Option<Pubkey> = metadata_pointer.metadata_address.into();
    // If metadata_address is set to the mint itself, metadata might already exist
    // Token-2022 will fail if we try to initialize twice, so we prevent it here
    if metadata_address.is_some() {
        // Note: This is a simplified check. In production, you might want to
        // check the actual metadata account data to see if it's initialized.
        // For now, we'll let the Token-2022 CPI fail with its own error if metadata exists.
        msg!("Warning: Metadata pointer is set. This mint may already have metadata.");
    }
}
```

However, the comments indicate that this check should be used to revert in the future. This leads to an issue as the `metadata_address` will always be `Some` as it is set when creating the mint by calling `metadata_pointer_instruction::initialize`.

```
// Initialize metadata pointer extension
let metadata_pointer_ix = metadata_pointer_instruction::initialize(
    &ctx.accounts.token_2022_program.key(),
    &ctx.accounts.mint.key(),
    Some(ctx.accounts.authority.key()),
    Some(ctx.accounts.mint.key()),
)?;

invoke(&metadata_pointer_ix, &[ctx.accounts.mint.to_account_info()])?;
```

As a result, the clause will always trigger and potentially block metadata creation in future versions.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/metadata.rs#L35-L46

Relevant Code

Mitigation Suggestion

We recommend removing the check as it will always be true.

Remediation

The issue has been fixed in `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by removing the metadata check.

ACC-I2 Both documentations are bloated and include the same content 2-4x

INFO WONTFIX

Description

For both programs, documentation is included in the `docs-content` folder. While there is a significant amount of text and four files per program, a large portion of the text across the files overlaps, and most of it is outdated. As a result, the duplicate content is sometimes read four or more times while reading through the documentation. Additionally, the files also include parts that do not fit with their intended content.

Location

https://github.com/tessera-lab/tessera-on-solana/tree/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/docs-content

Relevant Code

Mitigation Suggestion

We recommend rewriting the documentation to reduce bloat and outdated information significantly.

Remediation

Issue was marked as won't fix.

ACC-I3 Incorrect documentation of `update_fee_manager` INFO WONTFIX

Description

The documentation of `update_fee_manager` states the following:

```
#### `update_fee_manager`
- **Authority**: Primary Authority
- **Validation**: Authority must match mint's transfer fee config authority
- **Behavior**:
  - Update: Sets new fee manager or threshold
  - Close: If both parameters are `None`, closes config account
- **Events**: `FeeManagerUpdated`, `FeeManagerConfigClosed`
- **Note**: Closing config means fees can only be withdrawn using native Token-2022 authority
```

However actually `new_withdrawal_threshold` can never be None as it is not an option and the function also still closes the account even if it is not None.

```
pub fn update_fee_manager(
    ctx: Context<UpdateFeeManager>,
    new_fee_manager: Option<Pubkey>,
    new_withdrawal_threshold: u64,
) -> Result<()> {
```

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/update_fee_manager.rs#L8-L63

Relevant Code

Mitigation Suggestion

We recommend adapting the documentation to describe the actual behavior of the code.

Remediation

Issue was marked as won't fix.

ACC-I4 Incorrect decimals enforced on mint

INFO FIXED

Description

The documentation states that newly deployed mints should only be able to have 0-9 decimals.

```
#### `initialize_mint_with_transfer_fee`
- **Authority**: Any signer
- **Validation**:
  - `decimals`: 0-9 (enforced)
  - `transfer_fee_basis_points`: 0-10,000 (enforced)
  - `maximum_fee`: any value (0 = no cap)
```

When a new mint is deployed, the `validate_decimals` function is used to ensure this.

```
/// Validates that decimals is within acceptable range (0-18)
pub fn validate_decimals(decimals: u8) -> Result<()> {
    require!(decimals <= MAX_DECIMALS, ErrorCode::InvalidDecimals);
    Ok(())
}
```

When looking at `MAX_DECIMALS` one can see that it is defined as 18 instead of 9

```
// Token configuration limits
pub const MAX_DECIMALS: u8 = 18; // Standard maximum for SPL tokens
```

This allows users to deploy mints with more decimals than intended as based on the documentation.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/helpers.rs#L16-L20

Relevant Code

Mitigation Suggestion

We recommend changing `MAX_DECIMALS` to 9

Remediation

Fixed in `73d59a8b643c81a32ad4bc09160a6bae3a9eea74` by updating the documentation.

ACC-I5 Mint creation can be DOSd

INFO WONTFIX

Description

In `initialize_mint_with_transfer_fee`, the mint is created using the `create_account` system instruction.

```
// Create mint account
let rent = Rent::get()?;
let lamports = rent.minimum_balance(space);
let create_account_ix = system_instruction::create_account(
    &ctx.accounts.authority.key(),
    &ctx.accounts.mint.key(),
    lamports,
    space as u64,
    &ctx.accounts.token_2022_program.key(),
);

invoke(
    &create_account_ix,
    &[
        ctx.accounts.authority.to_account_info(),
        ctx.accounts.mint.to_account_info(),
        ctx.accounts.system_program.to_account_info(),
    ],
)?;
```

Unfortunately, this function has a bug that allows an attacker to block all account creation through it. Looking at the implementation of `create_account`, one can see that it returns an error if the account holds any lamports.

```
fn create_account(
    from_account_index: IndexOfAccount,
    to_account_index: IndexOfAccount,
    to_address: &Address,
    lamports: u64,
    space: u64,
    owner: &Pubkey,
    signers: &HashSet<Pubkey>,
    invoke_context: &InvokeContext,
    transaction_context: &TransactionContext,
    instruction_context: &InstructionContext,
) -> Result<(), InstructionError> {
    // if it looks like the `to` account is already in use, bail
    {
        let mut to = instruction_context
            .try_borrow_instruction_account(transaction_context, to_account_index)?;
        if to.get_lamports() > 0 {
            ic_msg!(
                invoke_context,
                "Create Account: account {:?} already in use",
                to_address
            );
            return Err(SystemError::AccountAlreadyInUse.into());
        }

        allocate_and_assign(&mut to, to_address, space, owner, signers, invoke_context)?;
    }
```

As a result, an attacker can block the mint's creation if they have prior knowledge of the address to which it will be deployed. This could be through public announcements or, if the deployer uses a private node, by an attacker monitoring traffic. In that case, the attacker can send the required rent for an account of size 0 to the account, and all creation attempts will be blocked.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/initialize_mint.rs#L47-L53

Relevant Code

Mitigation Suggestion

We recommend using `assign`, `transfer` and `allocate` instead of `create_account`.

Remediation

Issue was marked as won't fix.

ACC-I6 Oversized `SenderFeeConfig` leads to unnecessary rent cost

INFO WONTFIX

Description

The `SenderFeeConfig` account is initialized with a size that allows the `fee_recipients` vector to include up to 10 entries for fee recipients.

```
/// Sender-specific fee configuration for multi-tier referral distribution
#[account]
pub struct SenderFeeConfig {
    pub sender: Pubkey,
    pub fee_recipients: Vec<FeeRecipientInfo>,
    pub bump: u8,
}

impl SenderFeeConfig {
    // Space calculation:
    // 32 (sender Pubkey) + 4 (Vec length prefix) +
    // (32 + 2) * MAX_FEE_RECIPIENTS (Pubkey + u16 per recipient) +
    // 1 (bump u8)
    pub const LEN: usize = 32 + 4 + (32 + 2) * crate::constants::MAX_FEE_RECIPIENTS + 1;
}
```

However, when looking at what gets put into this vector (the `fee_recipients` returned by `build_fee_recipients()`), we can see that the maximum number of fee recipients that get pushed into this vector is 4.

1. Tier 1 fee recipient 2. Tier 2 fee recipient 3. Tier 3 fee recipient 4. Default fee recipient

So, initializing the account for a vector with 10 entries is a waste of space and leads to an unnecessary rent cost of 20c per account.

```
solana rent 181
Rent-exempt minimum: 0.00215064 SOL

solana rent 385
Rent-exempt minimum: 0.00357048 SOL

// Difference = 0.00141 SOL -> 0.00141 * $144 -> $0.20
```

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/state/accounts.rs#L23

Relevant Code

Mitigation Suggestion

We recommend adapting the constant to this:

```
/// Maximum number of fee recipients
pub const MAX_FEE_RECIPIENTS: usize = 4;
```

Remediation

Issue was marked as won't fix.

ACC-I7 Admin may close a referral code and user may reinitialize it

INFO FIXED

Description

We found that an admin may close a referral code, and then another user may reinitialize a new referral code at the same address or referral code of the closed code. This may lead to unintended consequences like incorrect referral counting.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/contexts/referral_code.rs#L60-L74

Relevant Code

```
/// referral_code.rs L60-L74
pub struct AdminCloseReferralCode<'info> {
    #[account(
        mut,
        seeds = [b"referral_code", referral_code.code.as_bytes()],
        bump = referral_code.bump,
        close = admin
    )]
    pub referral_code: Account<'info, ReferralCode>,

    #[account(seeds = [b"admin_list"], bump = admin_list.bump)]
    pub admin_list: Account<'info, AdminList>,

    #[account(mut)]
    pub admin: Signer<'info>,
}
```

Mitigation Suggestion

One option is to include the referral code owner's pubkey in the referral code pda derivation address. Another is to not allow closure of referral codes after they have been used already.

Remediation

Fixed by adding the owner key to the referral PDA `73d59a8b643c81a32ad4bc09160a6bae3a9eea74`.

ACC-I8 Referrer tier list can be overwritten and contains double record

INFO FIXED

Description

We found that the Referral chain assigns referral code owner to tier1 and tier2 as a fallback if no referrer user registration was passed. If re-registration occurs, the original tier 1 referrer is lost.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/helpers.rs#L110-L116

Relevant Code

```
/// helpers.rs L110-L116
            user_registration.tier2_referrer = referral_code_owner;
            user_registration.tier3_referrer = Pubkey::default();
        }
    } else {
        user_registration.tier2_referrer = referral_code_owner;
        user_registration.tier3_referrer = Pubkey::default();
    }
```

Mitigation Suggestion

By default, move all referrers on tier down and overwrite tier 2 and tier 3 only when a referral user registration was passed.

Example:

```
    user_registration.tier3_referrer = user_registration.tier2_referrer;
    user_registration.tier2_referrer = user_registration.tier1_referrer;
    user_registration.tier1_referrer = referral_code_owner;

    if let Some(referrer) = referrer_registration {
        if referrer.is_active {
            // Check for circular references: user cannot appear in their own chain
            require!(
                referrer.tier1_referrer != user_pubkey
                    && referrer.tier2_referrer != user_pubkey
                    && referrer.tier3_referrer != user_pubkey,
                ErrorCode::CannotUseSelfReferralCode
            );

            // Shift the chain: new user becomes tier1, others move down
            user_registration.tier2_referrer = referrer.tier1_referrer;
            user_registration.tier3_referrer = referrer.tier2_referrer;
        }
    }
```

Remediation

Remediation in 73d59a8b643c81a32ad4bc09160a6bae3a9eea74

ACC-I9 Modification after Validation in Create Referral Code

INFO WONTFIX

Description

We found that during the creation of a referral code, the program first validates constraints of the code and then "normalizes" it by uppercasing the code string. However, with UTF-8, codepoints do not always have their uppercase characters in the same byte stride as ASCII characters. For instance, the Greek character iota " ■ " codepoint "\u0390" expands from 2 bytes in its lowercase variant to 6 bytes in its upper case variant. The impact of this is largely mitigated by Anchor not being able to serialize the struct back into the Account, which will cause the transaction to fail.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/referral-system/src/instructions/referral_code.rs#L8-L12

Relevant Code

```
/// referral_code.rs L8-L12
    validate_referral_code(&code)?;

    // Normalize to uppercase for case-insensitive matching
    let normalized_code = normalize_referral_code(&code);
```

Mitigation Suggestion

Validate the referral code after the permutation takes place.

Remediation

Issue was marked as won't fix.

ACC-I10 FeeManager ATA is not owner validated

INFO WONTFIX

Description

We found that during the `withdraw_to_fee_manager`, the `fee_manager` ATA is not ownership validated. While the recalculation will produce the correct ATA, the `fee_manager` could have transferred the ownership of the token account to someone else. This someone else would then receive the fees. This could for example happen in either an offchain attack or a malicious fee manager trying to steal fess.

Location

https://github.com/tessera-lab/tessera-on-solana/blob/8ff55d1c1a544fed3fb099c271e49b40a4e0b5a8/programs/tessera-token/src/instructions/withdraw_fees.rs#L83-L98

Relevant Code

```
/// withdraw_fees.rs L83-L98
    let fee_manager_ata = get_associated_token_address_with_program_id(
        &config.fee_manager,
        &mint_key,
        &ctx.accounts.token_2022_program.key(),
    );

    require!(
        fee_manager_ata == ctx.accounts.fee_manager_ata.key(),
        ErrorCode::InvalidRecipient
    );

    // Check that the ATA account exists (has data)
    require!(
        ctx.accounts.fee_manager_ata.data_len() > 0,
        ErrorCode::FeeManagerAtaNotFound
    );
```

Mitigation Suggestion

Vaidate the ATA to belong to the `fee_manager`

Remediation

Issue was marked as won't fix.

APPENDIX

Vulnerability Classification

We rate our issues according to the following scale. Informational issues are reported informally to the developers and are not included in this report.

Severity	Description
Critical	Vulnerabilities that can be easily exploited and result in loss of user funds, or directly violate the protocol's integrity. Immediate action is required.
High	Vulnerabilities that can lead to loss of user funds under non-trivial preconditions, loss of fees, or permanent denial of service that requires a program upgrade. These issues require attention and should be resolved in the short term.
Medium	Vulnerabilities that may be more difficult to exploit but could still lead to some compromise of the system's functionality. For example, partial denial of service attacks, or such attacks that do not require a program upgrade to resolve, but may require manual intervention. These issues should be addressed as part of the normal development cycle.
Low	Vulnerabilities that have a minimal impact on the system's operations and can be fixed over time. These issues may include inconsistencies in state, or require such high capital investments that they are not exploitable profitably.
Informational	Findings that do not pose an immediate risk but could affect the system's efficiency, maintainability, or best practices.

Audit Methodology

Accretion is a boutique security auditor specializing in Solana's ecosystem. We employ a customized approach for each client, strategically allocating our resources to maximize code review effectiveness. Our auditors dedicate substantial time to developing a comprehensive understanding of each program under review, examining design decisions, expected and edge-case behaviors, invariants, optimizations, and data structures, while meticulously verifying mathematical correctness—all within the context of the developers' intentions.

Our audit scope extends beyond on-chain components to include associated infrastructure, such as user interfaces and supporting systems. Every audit encompasses both a holistic protocol design review and detailed line-by-line code analysis.

During our assessment, we focus on identifying:
• Solana-specific vulnerabilities
• Access control issues
• Arithmetic errors and precision loss
• Race conditions and MEV opportunities
• Logic errors and edge cases
• Performance optimization opportunities
• Invariant violations
• Account confusion vulnerabilities
• Authority check omissions
• Token22 implementation risks and SPL-related pitfalls
• Deviations from best practices

Our approach transcends conventional vulnerability classifications. We continuously conduct ecosystem-wide security research to identify and mitigate emerging threat vectors, ensuring our audits remain at the forefront of Solana security practices.