Safe Edges

AUDIT REPORT

2025

SAFEEDGES.IN

Count

Severity	Count	Status
Critical		
High		
Medium	4	Acknowledge
Low	1	Acknowledge
Info		
Gas		
Total		

Findings

M-01 User Balance Is Not Recorded During Deposit

Severity:

Medium

Description:
The `deposit` function transfers SOL from the user to the vault but does not update any on-chain record of the user's deposited amount. Without tracking user balances, the system cannot verify ownership or enforce limits during future withdrawals or interactions, leading to loss of accounting integrity.

```
pub fn deposit(ctx: Context<Deposit>, amount: u64, recipient: Vec<u8>) -> Result<()>
{
        let vault = &mut ctx.accounts.vault;
        let user = &mut ctx.accounts.user;

        // Transfer SOL from user to vault
        let cpi_context = CpiContext::new(
            ctx.accounts.system_program.to_account_info(),
            anchor_lang::system_program::Transfer {
                from: user.to_account_info(),
                to: vault.to_account_info(),
            },
        );
@>      anchor_lang::system_program::transfer(cpi_context, amount)?;

        msg!("Deposited {} lamports to {}", amount, hex::encode(&recipient));
        Ok(())
    }
```

Impact:
Users who deposit SOL have no on-chain proof of deposit, making refunding or withdrawal logic unreliable or insecure.

Recommendation:
Record the deposited amount in a user balance field after the transfer.

```
let user = &mut ctx.accounts.user;
+ user.balance += amount;
```

Team Response:

Acknowledge

M-02 Unauthorized User Can Front-Run Vault Initialization

Severity:

Medium

Description:
The `initialize` and function sets the `vault.authority` and assigns relayers without verifying that the caller is an authorized signer. Although it configures sensitive roles like relayers and the authority, it lacks any `signer` constraint on the `authority` account. This allows any user to front-run the initialize call and assign themselves or malicious actors as relayers and authority, especially in scenarios where the vault PDA is publicly derivable.

Impact:
An attacker can initialize the vault ahead of the legitimate user and seize control by setting themselves as authority and relayers, compromising the integrity of subsequent operations.

Recommendation:
Add a signer validation with the deployer public address and change them if required.

Team Response:

N/A

M-03 Unauthorized User Can Front-Run Vesting Initialization

Severity:

Medium

Description:
The `initialize_vesting` function sets critical state such as `vesting.creator` and `beneficiary` but does not require the creator to be a `signer`. As a result, a malicious actor can front-run the intended initializer and configure vesting parameters for arbitrary recipients, assuming they have sufficient token balance for the transfer.

Impact:
An attacker can seize control over a vesting contract by initializing it before the legitimate user, redirecting funds or vesting rights.

Recommendation:
Require deployer to be a signer to ensure only the intended party can initialize the vesting contract.

Team Response:

Acknowledge

M-04 Invalid Schedule Can Bypass Vesting Time Constraints

Severity:

Medium

Description:
The `initialize_vesting` function verifies that `end_ts > start_ts`, but does not ensure that `start_ts` lies in the future relative to the current block time. This allows initialization of vesting schedules with `start_ts` in the past, effectively bypassing the intended lock-in period and possibly enabling immediate or retroactive vesting.

```
pub fn initialize_vesting(
        ctx: Context<InitializeVesting>,
        amount: u64,
        start_ts: i64,
        cliff_amount: u64,
        end_ts: i64,
    ) -> Result<()> {
        let vesting = &mut ctx.accounts.vesting;

        require!(end_ts > start_ts, VestingError::InvalidSchedule);
        require!(amount > 0, VestingError::InvalidAmount);

...
}
```

Impact:
A vesting contract can be initialized with a schedule that appears already elapsed, causing premature token release and defeating time-lock assumptions.

Recommendation:
Add a constraint to validate that `start_ts` is greater than the current Unix timestamp.

Team Response:

Acknowledge

L-01 Missing Default Address Valdiation

Severity:

Low

Description:
In the `InitializeVesting` struct, `beneficiary` account is not checked for being the default zero address. This allows initializing a vesting schedule with an invalid or unclaimable beneficiary.

Impact:
Tokens could be locked in a vesting schedule with no way for anyone to claim them.

Recommendation:
Add a check to ensure the `beneficiary` is not the default address.

Team Response:

Acknowledge

About SAFE EDGES

SafeEdges is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

500+
Audits Completed

$3B
Secured

600k+
Lines of Code Audited