FYEO

Secure Code Review of Alliance

Terraform Labs

March 2023
Version 1.0

Presented by:

FYEO Inc.
PO Box 147044
Lakewood CO 80214
United States

Security Level
Strictly Confidential

TABLE OF CONTENTS

List of Figures

List of Tables

EXECUTIVE SUMMARY

OVERVIEW

Terraform Labs engaged FYEO Inc. to perform a Secure Code Review of Alliance.

The assessment was conducted remotely by the FYEO Security Team. Testing took place on January 26 - February 21, 2023, and focused on the following objectives:

- To provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the results of our tests.

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the FYEO Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

KEY FINDINGS

The following issues have been identified during the testing period. These should be prioritized for remediation to reduce the risk they pose:

- FYEO-TFL-01 – Division by zero in RebalanceBondTokenWeights
- FYEO-TFL-02 – Negative coin amount when validating delegated amount
- FYEO-TFL-03 – RewardChangeInterval can be negative
- FYEO-TFL-04 – There should be a minimum amount for Staking / Delegation operations
- FYEO-TFL-05 – Use of time.Now() can lead to consensus halt
- FYEO-TFL-06 – Users can un-delegate more tokens than expected
- FYEO-TFL-07 – Validator dust shares are not cleared
- FYEO-TFL-08 – Value assigned and immediately overwritten
- FYEO-TFL-09 – Variable shadowing means all errors are ignored

- FYEO-TFL-10 – InitGenesis and CreateAlliance do not ValidateDenom
- FYEO-TFL-11 – RebalanceBondTokenWeights returns success on error in MintCoins
- FYEO-TFL-12 – RewardWeightChangeHook function ignores possible error in UpdateAllianceAsset
- FYEO-TFL-13 – Calculation in for loop yields same result in each iteration
- FYEO-TFL-14 – Calculations done with amounts that are possibly 0
- FYEO-TFL-15 – Comments refer to the alliance module as the staking module
- FYEO-TFL-16 – DeleteAsset, ResetAssetAndValidators function should not rely on caller to check conditions
- FYEO-TFL-17 – Functions are identical
- FYEO-TFL-18 – Inconsistent GetDelegation / SetDelegation parameter use
- FYEO-TFL-19 – Use of Deprecated Function EmitEvents
- FYEO-TFL-20 – fraction is not validated in SlashValidator

Based on our review process, we conclude that the reviewed code implements the documented functionality.

SCOPE AND RULES OF ENGAGEMENT

The FYEO Review Team performed a Secure Code Review of Alliance. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

The source code was supplied through a public repository at https://github.com/terra-money/alliance with the commit hash 846d9a587080ac88237dda735b0f1995895d8203.

A re-review of the findings was performed on March 7, 2023, with the commit hash 0cbb2dafd8464ef8b968ef66d06978a7424c7786.

Files included in the code review

```
alliance-contracts/
├── app/
│   ├── app.go
│   ├── export.go
│   ├── genesis.go
│   ├── simulation_test.go
│   └── test_helpers.go
├── cmd/
│   └── allianced/
│       ├── main.go
│       └── testnet.go
├── custom/
│   └── bank/
│       ├── keeper/
│       │   └── keeper.go
│       ├── types/
│       │   └── keeper_interfaces.go
│       └── module.go
├── docs/
│   └── docs.go
├── testutil/
│   ├── keeper/
│   │   └── mocks.go
│   ├── network/
│   │   └── network.go
│   ├── nullify/
│   │   └── nullify.go
│   └── sample/
│       └── sample.go
└── x/
```

Files included in the code review

```
    └── alliance/
        ├── benchmark/
        │   ├── benchmark_test.go
        │   └── test_helper.go
        ├── client/
        │   ├── cli/
        │   │   ├── gov.go
        │   │   ├── query.go
        │   │   └── tx.go
        │   └── proposal_handler.go
        ├── e2e/
        │   ├── delegate_undelegate_test.go
        │   └── test_helper.go
        ├── keeper/
        │   ├── asset.go
        │   ├── asset_test.go
        │   ├── delegation.go
        │   ├── delegation_test.go
        │   ├── genesis.go
        │   ├── genesis_test.go
        │   ├── grpc_query.go
        │   ├── grpc_query_test.go
        │   ├── hooks.go
        │   ├── keeper.go
        │   ├── keeper_test.go
        │   ├── msg_server.go
        │   ├── params.go
        │   ├── proposal.go
        │   ├── proposal_test.go
        │   ├── reward.go
        │   ├── reward_test.go
        │   ├── slash.go
        │   ├── slash_test.go
        │   └── validator.go
        ├── simulation/
        │   ├── decoder.go
        │   ├── genesis.go
        │   ├── operations.go
        │   └── params.go
        ├── types/
        │   ├── alliance.go
        │   ├── alliance.pb.go
        │   ├── asset.go
```

Files included in the code review
```
│       ├── codec.go
│       ├── delegations.go
│       ├── delegations.pb.go
│       ├── errors.go
│       ├── events.go
│       ├── genesis.pb.go
│       ├── gov.go
│       ├── gov.pb.go
│       ├── keeper_interfaces.go
│       ├── keys.go
│       ├── keys_test.go
│       ├── msg.go
│       ├── params.go
│       ├── params.pb.go
│       ├── query.pb.go
│       ├── query.pb.gw.go
│       ├── tx.pb.go
│       ├── types_test.go
│       ├── validator.go
│       └── validator_test.go
├── abci.go
├── genesis.go
├── invariants.go
├── module.go
└── proposal_handler.go
```

Table 1: Scope

TECHNICAL ANALYSES AND FINDINGS

During the Secure Code Review of Alliance, we discovered:

- 9 findings with HIGH severity rating.
- 3 findings with LOW severity rating.
- 8 findings with INFORMATIONAL severity rating.

The following chart displays the findings by severity.

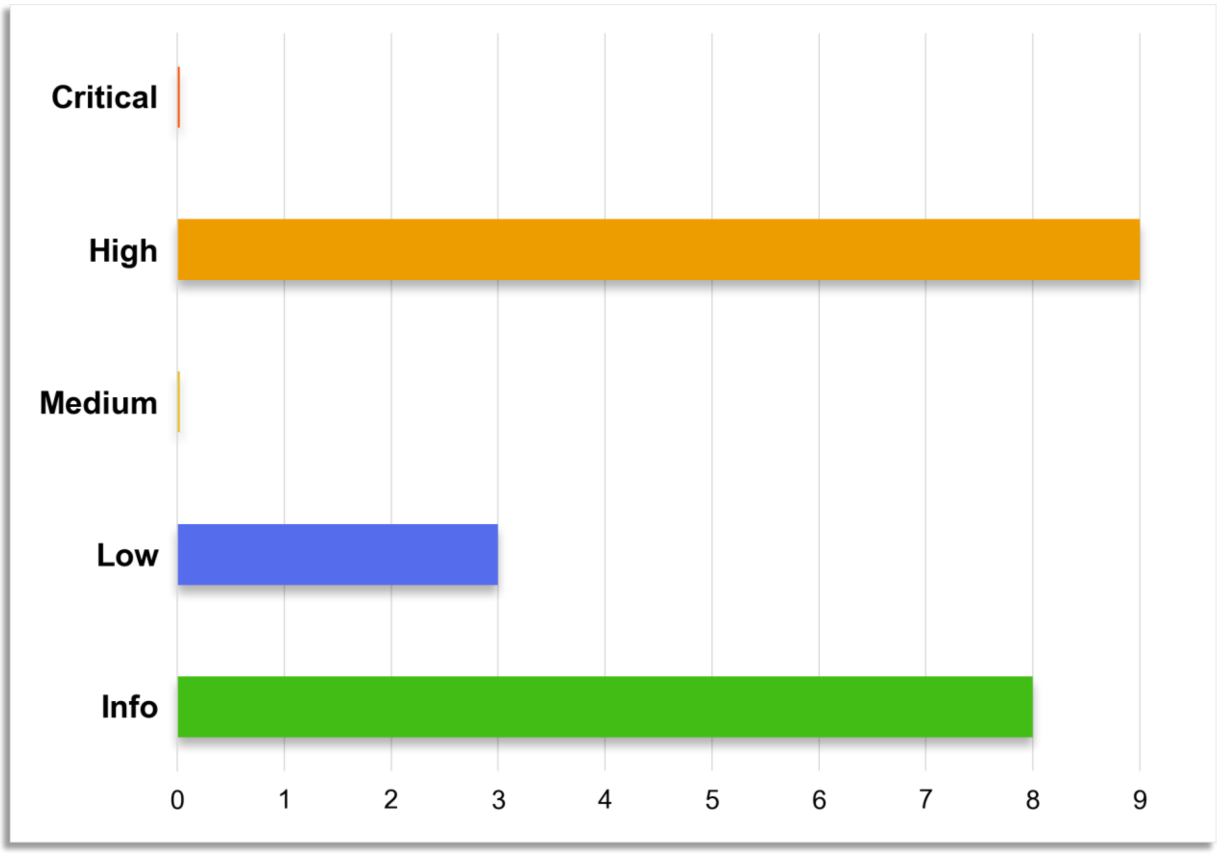

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

Finding #	Severity	Description
FYEO-TFL-01	High	Division by zero in RebalanceBondTokenWeights
FYEO-TFL-02	High	Negative coin amount when validating delegated amount
FYEO-TFL-03	High	RewardChangeInterval can be negative
FYEO-TFL-04	High	There should be a minimum amount for Staking / Delegation operations
FYEO-TFL-05	High	Use of time.Now() can lead to consensus halt
FYEO-TFL-06	High	Users can un-delegate more tokens than expected
FYEO-TFL-07	High	Validator dust shares are not cleared
FYEO-TFL-08	High	Value assigned and immediately overwritten
FYEO-TFL-09	High	Variable shadowing means all errors are ignored
FYEO-TFL-10	Low	InitGenesis and CreateAlliance do not ValidateDenom
FYEO-TFL-11	Low	RebalanceBondTokenWeights returns success on error in MintCoins
FYEO-TFL-12	Low	RewardWeightChangeHook function ignores possible error in UpdateAllianceAsset
FYEO-TFL-13	Informational	Calculation in for loop yields same result in each iteration
FYEO-TFL-14	Informational	Calculations done with amounts that are possibly 0
FYEO-TFL-15	Informational	Comments refer to the alliance module as the staking module
FYEO-TFL-16	Informational	DeleteAsset, ResetAssetAndValidators function should not rely on caller to check conditions

FYEO-TFL-17	**Informational**	Functions are identical
FYEO-TFL-18	**Informational**	Inconsistent GetDelegation / SetDelegation parameter use
FYEO-TFL-19	**Informational**	Use of Deprecated Function EmitEvents
FYEO-TFL-20	**Informational**	fraction is not validated in SlashValidator

Table 2: Findings Overview

TECHNICAL ANALYSIS

The source code has been manually validated to the extent that the state of the repository allowed. The validation includes confirming that the code correctly implements the intended functionality.

CONCLUSION

Based on our review process, we conclude that the code implements the documented functionality to the extent of the reviewed code.

Technical Findings

General Observations

It is evident that the go code being reviewed has been crafted with care and attention to detail. The code comments were concise and clear, which is a great foundation for easy understanding of the code. The tests were also well thought out and had good coverage, although there were some areas where the tests could be improved by using random inputs, which may have helped identify some problems that arose when certain more unusual values, such as tiny amounts, were used. This would be a helpful addition to an already solid test suite, ensuring a more comprehensive codebase.

The development team was quick to responded to our feedback and resolved the issues with great communication and at a rapid pace. The team's ability to address these issues quickly was a testament to their dedication to producing high-quality code. While there were some areas for improvement, overall, the code was well-structured and reasonably easy to read.

Division by zero in RebalanceBondTokenWeights

Finding ID: FYEO-TFL-01
Severity: High
Status: Remediated

Description

Alliance module panicked (division by zero) when users delegate with a small delegation amount and the `bondAmount` is 0.

Proof of Issue

Test case

```
   func TestPanic(t *testing.T) {
   app, ctx := createTestContext(t)
   startTime := time.Now()
   ctx = ctx.WithBlockTime(startTime).WithBlockHeight(1)
   app.AllianceKeeper.InitGenesis(ctx, &types.GenesisState{
      Params: types.DefaultParams(),
      Assets: []types.AllianceAsset{
         types.NewAllianceAsset(ALLIANCE_TOKEN_DENOM, sdk.NewDec(2), sdk.NewDec(0),
ctx.BlockTime()),
         types.NewAllianceAsset(ALLIANCE_2_TOKEN_DENOM, sdk.NewDec(10),
sdk.MustNewDecFromStr("0.1"), ctx.BlockTime()),
      },
   })
   queryServer := keeper.NewQueryServerImpl(app.AllianceKeeper)

   // Set tax and rewards to be zero for easier calculation
   distParams := app.DistrKeeper.GetParams(ctx)
   distParams.CommunityTax = sdk.ZeroDec()
   distParams.BaseProposerReward = sdk.ZeroDec()
   distParams.BonusProposerReward = sdk.ZeroDec()
   app.DistrKeeper.SetParams(ctx, distParams)

   // Accounts
   //mintPoolAddr := app.AccountKeeper.GetModuleAddress(minttypes.ModuleName)
   //rewardsPoolAddr := app.AccountKeeper.GetModuleAddress(types.RewardsPoolName)
   addrs := test_helpers.AddTestAddrsIncremental(app, ctx, 4, sdk.NewCoins(
      sdk.NewCoin(ALLIANCE_TOKEN_DENOM, sdk.NewInt(1000_000_000)),
      sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(2000_000_000)),
   ))
   pks := test_helpers.CreateTestPubKeys(2)

   // Creating two validators: 1 with 0% commission, 1 with 100% commission
   valAddr1 := sdk.ValAddress(addrs[0])
   _val1 := teststaking.NewValidator(t, valAddr1, pks[0])
   _val1.Commission = stakingtypes.Commission{
      CommissionRates: stakingtypes.CommissionRates{
         Rate:          sdk.NewDec(0),
         MaxRate:       sdk.NewDec(0),
         MaxChangeRate: sdk.NewDec(0),
      },
      UpdateTime: time.Now(),
   }
   test_helpers.RegisterNewValidator(t, app, ctx, _val1)
   val1, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr1)
   require.NoError(t, err)

   valAddr2 := sdk.ValAddress(addrs[1])
   _val2 := teststaking.NewValidator(t, valAddr2, pks[1])
   _val2.Commission = stakingtypes.Commission{
      CommissionRates: stakingtypes.CommissionRates{
         Rate:          sdk.NewDec(1),
         MaxRate:       sdk.NewDec(1),
         MaxChangeRate: sdk.NewDec(0),
      },
      UpdateTime: time.Now(),
   }
   test_helpers.RegisterNewValidator(t, app, ctx, _val2)
   val2, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr2)
   require.NoError(t, err)

   user1 := addrs[2]
   user2 := addrs[3]

   // Delegate token with non-zero take_rate
   _, err = app.AllianceKeeper.Delegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(100)))
   require.NoError(t, err)
   _, err = app.AllianceKeeper.Delegate(ctx, user2, val2,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(1000_000_000)))
   require.NoError(t, err)



   assets := app.AllianceKeeper.GetAllAssets(ctx)
   err = app.AllianceKeeper.RebalanceBondTokenWeights(ctx, assets)
   require.NoError(t, err)

   // Check total bonded amount
   //require.Equal(t, sdk.NewInt(11_000_000),
app.StakingKeeper.TotalBondedTokens(ctx))

   //ctx = ctx.WithBlockTime(startTime.Add(time.Minute * 6)).WithBlockHeight(2)
   coins, err := app.AllianceKeeper.DeductAssetsHook(ctx, assets)
   //require.NoError(t, err)
   //require.False(t, coins.IsZero())

   res, err := queryServer.AllianceDelegation(ctx,
&types.QueryAllianceDelegationRequest{
      DelegatorAddr: user1.String(),
      ValidatorAddr: val1.GetOperator().String(),
      Denom:         ALLIANCE_2_TOKEN_DENOM,
      Pagination:    nil,
   })
   require.NoError(t, err)
   del := res.GetDelegation()
   //require.True(t, del.GetBalance().Amount.LT(sdk.NewInt(1000_000_000)), "%s should
be less than %s", del.GetBalance().Amount, sdk.NewInt(1000_000_000))
   // Undelegate token with initial amount should fail
   _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(1000_000_000)))
   require.Error(t, err)

    fmt.Println("CURRENT BALANCE", del.Balance.Amount)

   // Undelegate token with current amount should pass
   _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(5001)))
   require.NoError(t, err)

   // User should have everything withdrawn
   _, found := app.AllianceKeeper.GetDelegation(ctx, user1, val1,
ALLIANCE_2_TOKEN_DENOM)
   require.False(t, found)

   // Delegate again
   _, err = app.AllianceKeeper.Delegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(500_000_000)))
   require.NoError(t, err)

   ctx = ctx.WithBlockTime(ctx.BlockTime().Add(time.Minute * 1)).WithBlockHeight(2)

   _, err = app.AllianceKeeper.Delegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(400_000_000)))
   require.NoError(t, err)

   ctx = ctx.WithBlockTime(ctx.BlockTime().Add(time.Minute * 5)).WithBlockHeight(3)
   coins, err = app.AllianceKeeper.DeductAssetsHook(ctx, assets)
   require.NoError(t, err)
   require.False(t, coins.IsZero())

   res, err = queryServer.AllianceDelegation(ctx,
&types.QueryAllianceDelegationRequest{
      DelegatorAddr: user1.String(),
      ValidatorAddr: val1.GetOperator().String(),
      Denom:         ALLIANCE_2_TOKEN_DENOM,
      Pagination:    nil,
   })
   require.NoError(t, err)
   del = res.GetDelegation()
   require.True(t, del.GetBalance().Amount.LT(sdk.NewInt(900_000_000)), "%s should be
less than %s", del.GetBalance().Amount, sdk.NewInt(1000_000_000))

   // Undelegate token with current amount should pass
   _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, del.Balance.Amount))
   require.NoError(t, err)

   // User should have everything withdrawn
   _, found = app.AllianceKeeper.GetDelegation(ctx, user1, val1,
ALLIANCE_2_TOKEN_DENOM)
   require.False(t, found)
   res, err = queryServer.AllianceDelegation(ctx,
&types.QueryAllianceDelegationRequest{
      DelegatorAddr: user1.String(),
      ValidatorAddr: val1.GetOperator().String(),
      Denom:         ALLIANCE_2_TOKEN_DENOM,
      Pagination:    nil,
   })
   require.NoError(t, err)
   del = res.GetDelegation()
   require.True(t, del.Balance.Amount.IsZero())
}
```

Severity and Impact Summary

Consensus and chain halt.

Recommendation

We recommend checking if the `bondAmount` is 0 or not.

NEGATIVE COIN AMOUNT WHEN VALIDATING DELEGATED AMOUNT

Finding ID: FYEO-TFL-02
Severity: High
Status: Remediated

Description

Alliance module panicked (negative coin amount) when users try to undelegate more amount than the delegated tokens. .

Proof of Issue

Test case: In this test case, the user1 delegated 5000 and wanted to undelegated 5001. The delegationSharesToUpdate is 5000.999999999999374875 and delegation.share is 5000.000000000000000000, and delegation.Shares.Sub(delegationSharesToUpdate) returned a negative amount and crashed the module.

```
   func TestNegativeCoin(t *testing.T) {
   app, ctx := createTestContext(t)
   startTime := time.Now()
   ctx = ctx.WithBlockTime(startTime).WithBlockHeight(1)
   app.AllianceKeeper.InitGenesis(ctx, &types.GenesisState{
      Params: types.DefaultParams(),
      Assets: []types.AllianceAsset{
         types.NewAllianceAsset(ALLIANCE_TOKEN_DENOM, sdk.NewDec(2), sdk.NewDec(0),
ctx.BlockTime()),
         types.NewAllianceAsset(ALLIANCE_2_TOKEN_DENOM, sdk.NewDec(10),
sdk.MustNewDecFromStr("0.1"), ctx.BlockTime()),
      },
   })
   queryServer := keeper.NewQueryServerImpl(app.AllianceKeeper)

   // Set tax and rewards to be zero for easier calculation
   distParams := app.DistrKeeper.GetParams(ctx)
   distParams.CommunityTax = sdk.ZeroDec()
   distParams.BaseProposerReward = sdk.ZeroDec()
   distParams.BonusProposerReward = sdk.ZeroDec()
   app.DistrKeeper.SetParams(ctx, distParams)

   // Accounts
   //mintPoolAddr := app.AccountKeeper.GetModuleAddress(minttypes.ModuleName)
   //rewardsPoolAddr := app.AccountKeeper.GetModuleAddress(types.RewardsPoolName)
   addrs := test_helpers.AddTestAddrsIncremental(app, ctx, 4, sdk.NewCoins(
      sdk.NewCoin(ALLIANCE_TOKEN_DENOM, sdk.NewInt(1000_000_000)),
      sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(2000_000_000)),
   ))
   pks := test_helpers.CreateTestPubKeys(2)

   // Creating two validators: 1 with 0% commission, 1 with 100% commission
   valAddr1 := sdk.ValAddress(addrs[0])
   _val1 := teststaking.NewValidator(t, valAddr1, pks[0])
   _val1.Commission = stakingtypes.Commission{
      CommissionRates: stakingtypes.CommissionRates{
         Rate:          sdk.NewDec(0),
         MaxRate:       sdk.NewDec(0),
         MaxChangeRate: sdk.NewDec(0),
      },
      UpdateTime: time.Now(),
   }
   test_helpers.RegisterNewValidator(t, app, ctx, _val1)
   val1, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr1)
   require.NoError(t, err)

   valAddr2 := sdk.ValAddress(addrs[1])
   _val2 := teststaking.NewValidator(t, valAddr2, pks[1])
   _val2.Commission = stakingtypes.Commission{
      CommissionRates: stakingtypes.CommissionRates{
         Rate:          sdk.NewDec(1),
         MaxRate:       sdk.NewDec(1),
         MaxChangeRate: sdk.NewDec(0),
      },
      UpdateTime: time.Now(),
   }
   test_helpers.RegisterNewValidator(t, app, ctx, _val2)
   val2, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr2)
   require.NoError(t, err)

   user1 := addrs[2]
   user2 := addrs[3]

   // Delegate token with non-zero take_rate
   _, err = app.AllianceKeeper.Delegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(5000)))
   require.NoError(t, err)
   _, err = app.AllianceKeeper.Delegate(ctx, user2, val2,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(1000_000_000)))
   require.NoError(t, err)



   assets := app.AllianceKeeper.GetAllAssets(ctx)
   err = app.AllianceKeeper.RebalanceBondTokenWeights(ctx, assets)
   require.NoError(t, err)

   // Check total bonded amount
   //require.Equal(t, sdk.NewInt(11_000_000),
app.StakingKeeper.TotalBondedTokens(ctx))

   ctx = ctx.WithBlockTime(startTime.Add(time.Minute * 6)).WithBlockHeight(2)
   //coins, err := app.AllianceKeeper.DeductAssetsHook(ctx, assets)
   //require.NoError(t, err)
   //require.False(t, coins.IsZero())

   res, err := queryServer.AllianceDelegation(ctx,
&types.QueryAllianceDelegationRequest{
      DelegatorAddr: user1.String(),
      ValidatorAddr: val1.GetOperator().String(),
      Denom:         ALLIANCE_2_TOKEN_DENOM,
      Pagination:    nil,
   })
   require.NoError(t, err)
   del := res.GetDelegation()
   //require.True(t, del.GetBalance().Amount.LT(sdk.NewInt(1000_000_000)), "%s should
be less than %s", del.GetBalance().Amount, sdk.NewInt(1000_000_000))
   // Undelegate token with initial amount should fail
   _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(1000_000_000)))
   require.Error(t, err)

    fmt.Println("CURRENT BALANCE", del.Balance.Amount)

   // Undelegate token with more than current amount still pass
   _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(5001)))
   require.NoError(t, err)

}
```

Severity and Impact Summary

Consensus and chain halt.

Recommendation

We recommend rounding up the calculated share when users undelegate tokens.

REWARDCHANGEINTERVAL CAN BE NEGATIVE

Finding ID: FYEO-TFL-03
Severity: High
Status: Remediated

Description

The `ValidateBasic` function for `CreateAlliance` and `UpdateAlliance` do not check `RewardChangeInterval` which can be a negative number.

Proof of Issue

File name: x/alliance/types/gov.go
Line number: 44

```
func (m *MsgCreateAllianceProposal) ValidateBasic() error {
... No checks for RewardChangeInterval
}
```

`Duration` is implemented as:

```
type Duration int64
```

The user value is then stored:

File name: x/alliance/keeper/proposal.go
Line number: 21

```
asset := types.AllianceAsset{
    Denom:                req.Denom,
    RewardWeight:         req.RewardWeight,
    TakeRate:             req.TakeRate,
    TotalTokens:          sdk.ZeroInt(),
    TotalValidatorShares: sdk.ZeroDec(),
    RewardStartTime:      rewardStartTime,
    RewardChangeRate:     req.RewardChangeRate,
    RewardChangeInterval: req.RewardChangeInterval,
    LastRewardChangeTime: rewardStartTime,
}
```

Severity and Impact Summary

A negative `RewardChangeInterval` set by either `CreateAlliance` or `UpdateAlliance` will completely break the reward calculations.

Recommendation

Make sure this can never be set to a negative value.

THERE SHOULD BE A MINIMUM AMOUNT FOR STAKING / DELEGATION OPERATIONS

Finding ID: FYEO-TFL-04
Severity: High
Status: Remediated

Description

Delegations can be done with any amount - including 1 token. Which, depending on the number of decimals, may be tiny. For redelegation for instance, one could redelegate individual tokens which would be added to the queue 1 token at a time - so adding hundreds of thousands of items to the queue which would then require processing that isn't going to be speedy. Undelegating 250.000 tokens individually in a test ran for 10 minutes before hitting a timeout panic.

Proof of Issue

File name: x/alliance/keeper/delegation.go
Line number: 59

```
func (k Keeper) Redelegate(ctx sdk.Context, delAddr sdk.AccAddress, srcVal
types.AllianceValidator, dstVal types.AllianceValidator, coin sdk.Coin) (*time.Time,
error) {
    if srcVal.Validator.Equal(dstVal.Validator) {
        return nil, status.Errorf(codes.InvalidArgument, "Cannot redelegate to the
same validator")
    }
...
func (k Keeper) Undelegate(ctx sdk.Context, delAddr sdk.AccAddress, validator
types.AllianceValidator, coin sdk.Coin) (*time.Time, error) {
    asset, found := k.GetAssetByDenom(ctx, coin.Denom)
    if !found {
        return nil, status.Errorf(codes.NotFound, "Asset with denom: %s does not
exist", coin.Denom)
    }
```

A test such as this fails after 10 minutes:

```
    for i < 250_000 {
        i += 1
        _, err = app.AllianceKeeper.Undelegate(ctx, delAddr, val,
sdk.NewCoin(ALLIANCE_TOKEN_DENOM, sdk.NewInt(1)))
        require.NoError(t, err)
    }
```

Severity and Impact Summary

These functions can be called with tiny token amounts and the test shows that the service will quickly run into timeout situation where processing takes more than 10 minutes.
These data are also inserted in queues which are used in the `EndBlocker` function called at the end of

each block. If the queue is sufficiently spammed with re/un-delegations this can become a DoS situation.

Recommendation

Make sure that amounts are sufficiently large or add a fee that would make such actions unaffordable.

References

> BeginBlocker and EndBlocker are a way for module developers to add automatic execution of logic to their module. This is a powerful tool that should be used carefully, as complex automatic functions can slow down or even halt the chain. - https://docs.cosmos.network/main/building-modules/beginblock-endblock

USE OF TIME.NOW() CAN LEAD TO CONSENSUS HALT

Finding ID: FYEO-TFL-05
Severity: High
Status: Remediated

Description

time.Now() is used in a lot of places (genesis, params, endblock). Local clock times are subjective and thus non-deterministic, and this can lead to consensus halt. This issue is similar to a high vulnerability in cosmos sdk that was recently disclosed. https://forum.cosmos.network/t/cosmos-sdk-vulnerability-retrospective-security-advisory-jackfruit-october-12-2021/5349 . We recommend using block time instead of time.Now()

Proof of Issue

File name: abci.go
Line number: 16-17

```
    func EndBlocker(ctx sdk.Context, k keeper.Keeper) []abci.ValidatorUpdate {
    defer telemetry.ModuleMeasureSince(types.ModuleName, time.Now(),
telemetry.MetricKeyEndBlocker)
```

File name: genesis.go
Line number: 29-33

```
    Params: types.Params{
            RewardDelayTime:        time.Hour * 24 * 7,
            TakeRateClaimInterval: time.Minute * 5,
            LastTakeRateClaimTime: time.Now(),
        },
```

File name: params.go
Line number: 49-53

```
    return Params{
        RewardDelayTime:       time.Hour * 24 * 7,
        TakeRateClaimInterval: time.Minute * 5,
        LastTakeRateClaimTime: time.Now(),
    }
```

Severity and Impact Summary

Consensus and chain halt.

Recommendation

We recommend using current block time instead of local computer's current time.

USERS CAN UN-DELEGATE MORE TOKENS THAN EXPECTED

Finding ID: FYEO-TFL-06
Severity: High
Status: Remediated

Description

Small rounding error when calculating the tokens using fixed point math when performing the division first that results in un-delegating more tokens than expected

Proof of Issue

File name: asset.go
Line number: 31-36

```
    func ConvertNewShareToDecToken(totalTokens sdk.Dec, totalShares sdk.Dec, shares sdk.Dec) (token sdk.Dec) {
    if totalShares.IsZero() {
        return totalTokens
    }
    return shares.Quo(totalShares).Mul(totalTokens)
}
```

Test case: In the following test case, the user1 delegated 200, when time move forward, his current balance is 199, but he was able to un-delegate 200 token.

```
 func TestUndelegate(t *testing.T) {
   app, ctx := createTestContext(t)
   startTime := time.Now()
   ctx = ctx.WithBlockTime(startTime).WithBlockHeight(1)
   app.AllianceKeeper.InitGenesis(ctx, &types.GenesisState{
      Params: types.DefaultParams(),
      Assets: []types.AllianceAsset{
         types.NewAllianceAsset(ALLIANCE_TOKEN_DENOM, sdk.NewDec(2), sdk.NewDec(0), ctx.BlockTime()),
         types.NewAllianceAsset(ALLIANCE_2_TOKEN_DENOM, sdk.NewDec(10), sdk.MustNewDecFromStr("0.1"), ctx.BlockTime()),
      },
   })
   queryServer := keeper.NewQueryServerImpl(app.AllianceKeeper)

   // Set tax and rewards to be zero for easier calculation
   distParams := app.DistrKeeper.GetParams(ctx)
   distParams.CommunityTax = sdk.ZeroDec()
   distParams.BaseProposerReward = sdk.ZeroDec()
   distParams.BonusProposerReward = sdk.ZeroDec()
   app.DistrKeeper.SetParams(ctx, distParams)

   // Accounts
   //mintPoolAddr := app.AccountKeeper.GetModuleAddress(minttypes.ModuleName)
   //rewardsPoolAddr := app.AccountKeeper.GetModuleAddress(types.RewardsPoolName)
   addrs := test_helpers.AddTestAddrsIncremental(app, ctx, 4, sdk.NewCoins(
      sdk.NewCoin(ALLIANCE_TOKEN_DENOM, sdk.NewInt(1000_000_000)),
      sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(2000_000_000)),
   ))
   pks := test_helpers.CreateTestPubKeys(2)

   // Creating two validators: 1 with 0% commission, 1 with 100% commission
   valAddr1 := sdk.ValAddress(addrs[0])
   _val1 := teststaking.NewValidator(t, valAddr1, pks[0])
   _val1.Commission = stakingtypes.Commission{
      CommissionRates: stakingtypes.CommissionRates{
         Rate:          sdk.NewDec(0),
         MaxRate:       sdk.NewDec(0),
         MaxChangeRate: sdk.NewDec(0),
      },
      UpdateTime: time.Now(),
   }
   test_helpers.RegisterNewValidator(t, app, ctx, _val1)
   val1, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr1)
   require.NoError(t, err)

   valAddr2 := sdk.ValAddress(addrs[1])
   _val2 := teststaking.NewValidator(t, valAddr2, pks[1])
   _val2.Commission = stakingtypes.Commission{
      CommissionRates: stakingtypes.CommissionRates{
         Rate:          sdk.NewDec(1),
         MaxRate:       sdk.NewDec(1),
         MaxChangeRate: sdk.NewDec(0),
      },
      UpdateTime: time.Now(),
   }
   test_helpers.RegisterNewValidator(t, app, ctx, _val2)
   val2, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr2)
   require.NoError(t, err)

   user1 := addrs[2]
   user2 := addrs[3]

   // Delegate token with non-zero take_rate
   _, err = app.AllianceKeeper.Delegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(200)))
   require.NoError(t, err)
   _, err = app.AllianceKeeper.Delegate(ctx, user2, val2,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(1000_000_000)))
   require.NoError(t, err)



   assets := app.AllianceKeeper.GetAllAssets(ctx)
   err = app.AllianceKeeper.RebalanceBondTokenWeights(ctx, assets)
   require.NoError(t, err)

   // Check total bonded amount
   //require.Equal(t, sdk.NewInt(11_000_000),
app.StakingKeeper.TotalBondedTokens(ctx))

   ctx = ctx.WithBlockTime(startTime.Add(time.Minute * 6)).WithBlockHeight(2)
   //coins, err := app.AllianceKeeper.DeductAssetsHook(ctx, assets)
   //require.NoError(t, err)
   //require.False(t, coins.IsZero())

   res, err := queryServer.AllianceDelegation(ctx,
&types.QueryAllianceDelegationRequest{
      DelegatorAddr: user1.String(),
      ValidatorAddr: val1.GetOperator().String(),
      Denom:         ALLIANCE_2_TOKEN_DENOM,
      Pagination:    nil,
   })
   require.NoError(t, err)
   del := res.GetDelegation()
   //require.True(t, del.GetBalance().Amount.LT(sdk.NewInt(1000_000_000)), "%s should
be less than %s", del.GetBalance().Amount, sdk.NewInt(1000_000_000))
   // Undelegate token with initial amount should fail
   _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(1000_000_000)))
   require.Error(t, err)

    fmt.Println("CURRENT BALANCE", del.Balance.Amount)

   // Undelegate token with more than current amount still pass
   _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
sdk.NewCoin(ALLIANCE_2_TOKEN_DENOM, sdk.NewInt(200)))
   require.NoError(t, err)

}
```

Severity and Impact Summary

Insolvency

Recommendation

We recommend rounding up the calculated share when users undelegate tokens.

VALIDATOR DUST SHARES ARE NOT CLEARED

Finding ID: FYEO-TFL-07
Severity: High
Status: Remediated

Description

Alliance module panicked (division by zero) when users delegate again after un-delegating all tokens because the process of clearing the validator dust shares was skipped when the delegation was deleted.

Proof of Issue

Test case: In this test case, we delegated 1000 token, then un-delegated current balance which is 900 tokens, and then it panicked when we delegate 1000 token again . After un-delegating, the total token is 0.000000000000000000, but the total share is 0.000000000000000100, Which is not zero, and this lead to div by zero panic in `ConvertNewTokenToShares` function

```
   func TestRoundingError(t *testing.T) {
   app, ctx := createTestContext(t)
   startTime := time.Now()
   ctx = ctx.WithBlockTime(startTime).WithBlockHeight(1)
   app.AllianceKeeper.InitGenesis(ctx, &types.GenesisState{
      Params: types.DefaultParams(),
      Assets: []types.AllianceAsset{
         types.NewAllianceAsset(AllianceDenom, sdk.NewDec(2), sdk.NewDec(0),
ctx.BlockTime()),
         types.NewAllianceAsset(AllianceDenomTwo, sdk.MustNewDecFromStr("10"),
sdk.MustNewDecFromStr("0.1"), ctx.BlockTime()),
      },
   })
   queryServer := keeper.NewQueryServerImpl(app.AllianceKeeper)

   // Set tax and rewards to be zero for easier calculation
   distParams := app.DistrKeeper.GetParams(ctx)
   distParams.CommunityTax = sdk.ZeroDec()
   distParams.BaseProposerReward = sdk.ZeroDec()
   distParams.BonusProposerReward = sdk.ZeroDec()
   app.DistrKeeper.SetParams(ctx, distParams)

   // Accounts

   // rewardsPoolAddr := app.AccountKeeper.GetModuleAddress(types.RewardsPoolName)
   addrs := test_helpers.AddTestAddrsIncremental(app, ctx, 4, sdk.NewCoins(
      sdk.NewCoin(AllianceDenom, sdk.NewInt(1000_000_000)),
      sdk.NewCoin(AllianceDenomTwo, sdk.NewInt(2000_000_000)),
   ))
   pks := test_helpers.CreateTestPubKeys(2)

   // Creating two validators: 1 with 0% commission, 1 with 100% commission
   valAddr1 := sdk.ValAddress(addrs[0])
   _val1 := teststaking.NewValidator(t, valAddr1, pks[0])
```

```
    _val1.Commission = stakingtypes.Commission{
       CommissionRates: stakingtypes.CommissionRates{
          Rate:          sdk.NewDec(0),
          MaxRate:       sdk.NewDec(0),
          MaxChangeRate: sdk.NewDec(0),
       },
       UpdateTime: time.Now(),
    }
    test_helpers.RegisterNewValidator(t, app, ctx, _val1)
    val1, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr1)
    require.NoError(t, err)

    valAddr2 := sdk.ValAddress(addrs[1])
    _val2 := teststaking.NewValidator(t, valAddr2, pks[1])
    _val2.Commission = stakingtypes.Commission{
       CommissionRates: stakingtypes.CommissionRates{
          Rate:          sdk.NewDec(1),
          MaxRate:       sdk.NewDec(1),
          MaxChangeRate: sdk.NewDec(0),
       },
       UpdateTime: time.Now(),
    }
    test_helpers.RegisterNewValidator(t, app, ctx, _val2)
    val2, err := app.AllianceKeeper.GetAllianceValidator(ctx, valAddr2)
    require.NoError(t, err)

    user1 := addrs[2]
    user2 := addrs[3]

    // Delegate token with non-zero take_rate
    _, err = app.AllianceKeeper.Delegate(ctx, user1, val1,
sdk.NewCoin(AllianceDenomTwo, sdk.NewInt(1000)))
    require.NoError(t, err)
    _, err = app.AllianceKeeper.Delegate(ctx, user2, val2,
sdk.NewCoin(AllianceDenomTwo, sdk.NewInt(0)))
    require.NoError(t, err)



    assets := app.AllianceKeeper.GetAllAssets(ctx)

    err = app.AllianceKeeper.RebalanceBondTokenWeights(ctx, assets)

    require.NoError(t, err)

    // Check total bonded amount
    //require.Equal(t, sdk.NewInt(11_000_000),
app.StakingKeeper.TotalBondedTokens(ctx))

    ctx = ctx.WithBlockTime(startTime.Add(time.Minute * 6)).WithBlockHeight(2)
    _, err = app.AllianceKeeper.DeductAssetsHook(ctx, assets)
    require.NoError(t, err)
    //require.False(t, coins.IsZero())

    res, err := queryServer.AllianceDelegation(ctx,
&types.QueryAllianceDelegationRequest{
       DelegatorAddr: user1.String(),
       ValidatorAddr: val1.GetOperator().String(),
       Denom:         AllianceDenomTwo,
       Pagination:    nil,
    })
    require.NoError(t, err)
    del := res.GetDelegation()

    fmt.Println("CURRENT BALANCE", del.Balance.Amount)
     //Undelegate token with current amount should pass
    _, err = app.AllianceKeeper.Undelegate(ctx, user1, val1,
 sdk.NewCoin(AllianceDenomTwo, del.Balance.Amount))
    require.NoError(t, err)

     assets = app.AllianceKeeper.GetAllAssets(ctx)

    err = app.AllianceKeeper.RebalanceBondTokenWeights(ctx, assets)

    require.NoError(t, err)

 _, err = app.AllianceKeeper.Delegate(ctx, user1, val1, sdk.NewCoin(AllianceDenomTwo,
 sdk.NewInt(200)))
    require.NoError(t, err)


 }
```

Severity and Impact Summary

Consensus and chain halt.

Recommendation

We recommend clearing the dust shares even if the delegations are already deleted.

VALUE ASSIGNED AND IMMEDIATELY OVERWRITTEN

Finding ID: FYEO-TFL-08
Severity: High
Status: Remediated

Description

The variable `asset.LastRewardChangeTime` is set inside the if block only to be immediately overwritten below.

Proof of Issue

File name: x/alliance/keeper/asset.go
Line number: 44

```
// If there was a change in reward decay rate or reward decay time
if !newAsset.RewardChangeRate.Equal(asset.RewardChangeRate) ||
newAsset.RewardChangeInterval != asset.RewardChangeInterval {
    // And if there were no reward changes scheduled previously, start the counter
from now
    if asset.RewardChangeRate.Equal(sdk.OneDec()) || asset.RewardChangeInterval == 0 {
        asset.LastRewardChangeTime = ctx.BlockTime()
    }
    ...
}

// Make sure only whitelisted fields can be updated
...
asset.LastRewardChangeTime = newAsset.LastRewardChangeTime
```

Severity and Impact Summary

The `LastRewardChangeTime` is meant to be assigned to the current block time. Instead the original `LastRewardChangeTime` is kept unchanged. This would mean that the reward change is likely going to be changed right away with the first call to `RewardWeightChangeHook` from `EndBLocker`. This would essentially skip over the first interval and alter the `asset.RewardWeight` and therefore the rewards. There is no way to modify or correct the `LastRewardChangeTime` by other means.

Recommendation

The value was probably meant to be assigned to `newAsset.LastRewardChangeTime`.

VARIABLE SHADOWING MEANS ALL ERRORS ARE IGNORED

Finding ID: FYEO-TFL-09
Severity: High
Status: Remediated

Description

In the UpdateAllianceAsset a callback is used to `IterateAllianceValidatorInfo` - all errors potentially encountered by this callback are ignored.

The `var err error` is created and later checked to see if `if err != nil`. It looks like the intention is to catch any error inside the callback. However, the errors inside the callback only make the wrapping function skip the rest and return early. Errors inside the callback are not assigned to the outside `err` variable. The declaration of this variable is shadowed. Meaning if there ever is an error when iterating over this callback, it will not propagate back and thus `if err != nil` will always find `err` to be `nil`.

Proof of Issue

File name: x/alliance/keeper/asset.go
Line number: 24

```
var err error
// Only add a snapshot if reward weight changes
if !newAsset.RewardWeight.Equal(asset.RewardWeight) {
    k.IterateAllianceValidatorInfo(ctx, func(valAddr sdk.ValAddress, info
types.AllianceValidatorInfo) bool {
        validator, err := k.GetAllianceValidator(ctx, valAddr)
        if err != nil {
            return true
        }
        _, err = k.ClaimValidatorRewards(ctx, validator)
        if err != nil {
            return true
        }
        k.SetRewardWeightChangeSnapshot(ctx, asset, validator)
        return false
    })
    if err != nil {
        return err
    }
    // Queue a re-balancing event if reward weight change
    k.QueueAssetRebalanceEvent(ctx)
}
```

Severity and Impact Summary

While an error seems unlikely to arise in the call to `ClaimValidatorRewards` it would still prevent the remaining validators from creating the `RewardWeightChangeSnapshot`s which would ultimately affect the reward calculations.

Recommendation

Do not re-declare the `err` variable in the callback, so that values are correctly assigned to the outside declaration of the `err` variable.

InitGenesis and CreateAlliance do not ValidateDenom

Finding ID: FYEO-TFL-10
Severity: Low
Status: Remediated

Description

`CreateAlliance` and `InitGenesis` do not check the assets denom using `ValidateDenom(coin.Denom)` and it is therefore possible to create assets with a 0 byte or with other invalid formats.

Proof of Issue

File name: x/alliance/keeper/proposal.go
Line number: 12

```
func (k Keeper) CreateAlliance(ctx context.Context, req
*types.MsgCreateAllianceProposal) error {
    sdkCtx := sdk.UnwrapSDKContext(ctx)
    _, found := k.GetAssetByDenom(sdkCtx, req.Denom)

    if found {
        return ...
    }

    rewardStartTime := sdkCtx.BlockTime().Add(k.RewardDelayTime(sdkCtx))
    asset := types.AllianceAsset{
        Denom:                  req.Denom,
        ...
    }
    k.SetAsset(sdkCtx, asset)
    return nil
}
```

In this test, no error is caused. The asset is stored as is.

```
var byteArray2 = []byte{'u', 'l', 'u', 'n', 'a', 0, '2'}
var DENOM_2 = string(byteArray2[:len(byteArray2)])

createErr := app.AllianceKeeper.CreateAlliance(ctx, &types.MsgCreateAllianceProposal{
    Title:        "",
    Description:  "",
    Denom:        DENOM_2,
    RewardWeight: sdk.OneDec(),
    TakeRate:     sdk.OneDec(),
})
```

Severity and Impact Summary

These assets are stored and since functions such as `sdk.NewCoin` do call `ValidateDenom(coin.Denom)`, this may end up causing panics which can make certain functions unusable.

The code in `CreateDenomAddressPrefix` creates 0 terminated keys to make a scanable index. These scans would run into issues since asset denoms can contain 0 bytes.

As can be seen attaching a 0 byte would leave a scan interpreting both as the same asset:

```
denom "alliance" -> bytes: [97 108 108 105 97 110 99 101]
denom "alliance{0x00}2" -> bytes: [97 108 108 105 97 110 99 101 0 50]
```

Recommendation

Make sure to not allow assets with invalid denoms to be created in either `CreateAlliance` or `InitGenesis`.

REBALANCEBONDTOKENWEIGHTS RETURNS SUCCESS ON ERROR IN MINTCOINS

Finding ID: FYEO-TFL-11
Severity: Low
Status: Remediated

Description

In case of an error in `MintCoins`, the `RebalanceBondTokenWeights` will return success instead of propagating the error.

Proof of Issue

File name: x/alliance/keeper/asset.go
Line number: 130

```
if expectedBondAmount.GT(currentBondedAmount) {
    // delegate more tokens to increase the weight
    bondAmount := expectedBondAmount.Sub(currentBondedAmount).TruncateInt()
    err = k.bankKeeper.MintCoins(ctx, types.ModuleName,
sdk.NewCoins(sdk.NewCoin(bondDenom, bondAmount)))
    if err != nil {
        return nil
    }
    _, err = k.stakingKeeper.Delegate(ctx, moduleAddr, bondAmount,
stakingtypes.Unbonded, *validator.Validator, true)
    if err != nil {
        return err
    }
}
```

Severity and Impact Summary

The likelihood of an error seems low. However this return is nested inside a loop that iterates over all validators. Returning early like this means that none of the remaining validators get processed.

Recommendation

Return the error.

RewardWeightChangeHook function ignores possible error in UpdateAllianceAsset

Finding ID: FYEO-TFL-12
Severity: Low
Status: Remediated

Description

In `func (k Keeper) RewardWeightChangeHook`, the function `k.UpdateAllianceAsset(ctx, *asset)` is called. This function can return an error but any error would be ignored.

Proof of Issue

File name: x/alliance/keeper/asset.go
Line number: 316

```
k.QueueAssetRebalanceEvent(ctx)
k.UpdateAllianceAsset(ctx, *asset)
```

Severity and Impact Summary

TODO

Recommendation

Handle the error.

CALCULATION IN FOR LOOP YIELDS SAME RESULT IN EACH ITERATION

Finding ID: FYEO-TFL-13
Severity: Informational
Status: Remediated

Description

The code calculates `delegationTokens` in a loop, yet this calculations inputs will never change. Therefore the same value is calculated in each iteration of the loop.
Also, there is currently no test with several assets in the history, the addition of such a test would make sure the calculations remain correct and benefit the test coverage.

Proof of Issue

File name: x/alliance/keeper/reward.go
Line number: 97

```
for _, history := range latestRewardHistories {
    rewardHistory, found := rewardHistories.GetIndexByDenom(history.Denom)
    if !found {
        rewardHistory.Denom = history.Denom
        rewardHistory.Index = sdk.ZeroDec()
    }
    if rewardHistory.Index.GTE(history.Index) {
        continue
    }
    delegationTokens := sdk.NewDecFromInt(types.GetDelegationTokens(delegation,
validator, asset).Amount)
    claimWeight := delegationTokens.Mul(rewardWeight)
    totalClaimable := (history.Index.Sub(rewardHistory.Index)).Mul(claimWeight)
    rewardHistory.Index = history.Index
    rewards = rewards.Add(sdk.NewCoin(history.Denom, totalClaimable.TruncateInt()))
    if !found {
        rewardHistories = append(rewardHistories, *rewardHistory)
    }
}
```

Severity and Impact Summary

No impact to security. It is simply wasting CPU cycles.

Recommendation

Calculate the value once. Add a test that covers the for loop.

CALCULATIONS DONE WITH AMOUNTS THAT ARE POSSIBLY 0

Finding ID: FYEO-TFL-14
Severity: Informational
Status: Remediated

Description

Both `bondAmount` and `unbondAmount` may be truncated to 0. In such case the function could `continue`.

Proof of Issue

File name: x/alliance/keeper/asset.go
Line number: 130

```
if expectedBondAmount.GT(currentBondedAmount) {
    // delegate more tokens to increase the weight
    bondAmount := expectedBondAmount.Sub(currentBondedAmount).TruncateInt()
    err = k.bankKeeper.MintCoins(ctx, types.ModuleName,
sdk.NewCoins(sdk.NewCoin(bondDenom, bondAmount)))
    if err != nil {
        return nil
    }
    _, err = k.stakingKeeper.Delegate(ctx, moduleAddr, bondAmount,
stakingtypes.Unbonded, *validator.Validator, true)
    if err != nil {
        return err
    }
} else if expectedBondAmount.LT(currentBondedAmount) {
    // undelegate more tokens to reduce the weight
    unbondAmount := currentBondedAmount.Sub(expectedBondAmount).TruncateInt()
    sharesToUnbond, err := k.stakingKeeper.ValidateUnbondAmount(ctx, moduleAddr,
validator.GetOperator(), unbondAmount)
    if err != nil {
        return err
    }
    tokensToBurn, err := k.stakingKeeper.Unbond(ctx, moduleAddr,
validator.GetOperator(), sharesToUnbond)
    if err != nil {
        return err
    }
    err = k.bankKeeper.BurnCoins(ctx, stakingtypes.BondedPoolName,
sdk.NewCoins(sdk.NewCoin(bondDenom, tokensToBurn)))
    if err != nil {
        return err
    }
}
```

Severity and Impact Summary

Optimization.

Recommendation

Consider using `continue` if the value is 0.

COMMENTS REFER TO THE ALLIANCE MODULE AS THE STAKING MODULE

Finding ID: FYEO-TFL-15
Severity: Informational
Status: Remediated

Description

Some comments refer to the alliance module as the staking module.

Proof of Issue

File name: x/alliance/types/keys.go
Line number: 10

```
// ModuleName is the name of the staking module
ModuleName = "alliance"

// RewardsPoolName is the name of the module account for rewards
RewardsPoolName = "alliance_rewards"

// StoreKey is the string store representation
StoreKey = ModuleName

// QuerierRoute is the querier route for the staking module
QuerierRoute = ModuleName

// RouterKey is the msg router key for the staking module
RouterKey = ModuleName

// MemStoreKey defines the in-memory store key
MemStoreKey = "mem_signletimemodule"
```

Severity and Impact Summary

Not a code issue.

Recommendation

Correct the comments.

DeleteAsset, ResetAssetAndValidators function should not rely on caller to check conditions

Finding ID: FYEO-TFL-16
Severity: Informational
Status: Remediated

Description

An asset is unconditionally deleted by the `DeleteAsset` and `ResetAssetAndValidators` functions. The `ResetAssetAndValidators` is also declared to return an error but doesn't actually do so. Finally, the caller does not handle the error.

Proof of Issue

File name: x/alliance/keeper/asset.go
Line number: 196

```
func (k Keeper) DeleteAsset(ctx sdk.Context, denom string) {
    store := ctx.KVStore(k.storeKey)
    assetKey := types.GetAssetKey(denom)
    store.Delete(assetKey)
}
```

File name: x/alliance/keeper/delegation.go
Line number: 177

```
if asset.TotalTokens.IsZero() {
    k.ResetAssetAndValidators(ctx, asset)
}
```

File name: x/alliance/keeper/delegation.go
Line number: 551

```
func (k Keeper) ResetAssetAndValidators(ctx sdk.Context, asset types.AllianceAsset)
(err error) {
    k.IterateAllianceValidatorInfo(ctx, func(valAddr sdk.ValAddress, info
types.AllianceValidatorInfo) (stop bool) {
        updatedShares := sdk.NewDecCoins()
        for _, share := range info.ValidatorShares {
            if share.Denom != asset.Denom {
                updatedShares = append(updatedShares, share)
            }
        }
        info.ValidatorShares = updatedShares
        k.SetValidatorInfo(ctx, valAddr, info)
        return false
    })
    asset.TotalValidatorShares = sdk.ZeroDec()
    k.SetAsset(ctx, asset)
    return nil
}
```

Severity and Impact Summary

The conditions for deletion are best checked in the function doing the deletion. Otherwise, when additional code is added, there is some chance of the caller not checking those conditions.

Recommendation

Move the checks and return errors when appropriate.

FUNCTIONS ARE IDENTICAL

Finding ID: FYEO-TFL-17
Severity: **Informational**
Status: **Remediated**

Description

These two functions do the same / return the same value pair wise.

Proof of Issue

File name: x/alliance/types/validator.go
Line number: 48

```
func (v AllianceValidator) TotalSharesWithDenom(denom string) sdk.Dec {
    return sdk.DecCoins(v.TotalDelegatorShares).AmountOf(denom)
}

func (v AllianceValidator) TotalDelegationSharesWithDenom(denom string) sdk.Dec {
    return sdk.DecCoins(v.TotalDelegatorShares).AmountOf(denom)
}
```

File name: x/alliance/types/validator.go
Line number: 66

```
func (v AllianceValidator) TotalTokensWithAsset(asset AllianceAsset) sdk.Dec {
    shares := v.ValidatorSharesWithDenom(asset.Denom)
    dec := ConvertNewShareToDecToken(sdk.NewDecFromInt(asset.TotalTokens),
asset.TotalValidatorShares, shares)
    return dec
}

func (v AllianceValidator) TotalDecTokensWithAsset(asset AllianceAsset) sdk.Dec {
    shares := v.ValidatorSharesWithDenom(asset.Denom)
    return ConvertNewShareToDecToken(sdk.NewDecFromInt(asset.TotalTokens),
asset.TotalValidatorShares, shares)
}
```

Severity and Impact Summary

Code duplication is a maintainability concern.

Recommendation

Keep only one version of each.

INCONSISTENT GETDELEGATION / SETDELEGATION PARAMETER USE

Finding ID: FYEO-TFL-18
Severity: Informational
Status: Remediated

Description

`SetDelegation` uses `ValAddress`, `GetDelegation` takes `types.AllianceValidator` but only to get its `ValAddress`.

Proof of Issue

File name: x/alliance/keeper/delegation.go
Line number: 249

```
func (k Keeper) GetDelegation(ctx sdk.Context, delAddr sdk.AccAddress, validator
types.AllianceValidator, denom string) (d types.Delegation, found bool) {
    key := types.GetDelegationKey(delAddr, validator.GetOperator(), denom)
    b := ctx.KVStore(k.storeKey).Get(key)
    if b == nil {
        return d, false
    }
    k.cdc.MustUnmarshal(b, &d)
    return d, true
}

func (k Keeper) SetDelegation(ctx sdk.Context, delAddr sdk.AccAddress, valAddr
sdk.ValAddress, denom string, del types.Delegation) {
    key := types.GetDelegationKey(delAddr, valAddr, denom)
    b := k.cdc.MustMarshal(&del)
    ctx.KVStore(k.storeKey).Set(key, b)
}
```

Severity and Impact Summary

No impact.

Recommendation

Could be made more consistent by only using the address as nothing else is required.

USE OF DEPRECATED FUNCTION EMITEVENTS

Finding ID: FYEO-TFL-19
Severity: Informational
Status: Remediated

Description

`EmitEvents` is deprecated and `EmitTypedEvents` is its replacement. There are 4 calls to `EmitEvents` in `msg_server.go`.

Proof of Issue

File name: x/alliance/keeper/msg_server.go
Line number: 43

```
sdkCtx.EventManager().EmitEvents(sdk.Events{
    sdk.NewEvent(
        types.EventTypeDelegate,
        sdk.NewAttribute(types.AttributeKeyValidator, msg.ValidatorAddress),
        sdk.NewAttribute(sdk.AttributeKeyAmount, msg.Amount.String()),
        sdk.NewAttribute(types.AttributeKeyNewShares, newShares.String()),
    ),
})
```

Severity and Impact Summary

No impact.

Recommendation

Consider switching to the non deprecated function.

FRACTION IS NOT VALIDATED IN SLASHVALIDATOR

Finding ID: FYEO-TFL-20
Severity: Informational
Status: Remediated

Description

A `fraction` of validator shares are slashed, but the input `fraction` is never validated.

Proof of Issue

File name: slash.go
Line number: 21-32

```
    for _, share := range val.ValidatorShares {
        sharesToSlash := share.Amount.Mul(fraction)
        sharesAfterSlashing := sdk.NewDecCoinFromDec(share.Denom,
share.Amount.Sub(sharesToSlash))
        slashedValidatorShares = slashedValidatorShares.Add(sharesAfterSlashing)
        asset, found := k.GetAssetByDenom(ctx, share.Denom)
        if !found {
            return types.ErrUnknownAsset
        }
        asset.TotalValidatorShares = asset.TotalValidatorShares.Sub(sharesToSlash)
        k.SetAsset(ctx, asset)
    }
```

Severity and Impact Summary

Incorrect slashing amount.

Recommendation

We recommend checking if `fraction` is in (0,1] range .

OUR PROCESS

METHODOLOGY

FYEO Inc. uses the following high-level methodology when approaching engagements. They are broken up into the following phases.

Kickoff
Ramp-up
Review
Report
Verify

Figure 2: Methodology Flow

KICKOFF

The project is kicked off as the sales process has concluded. We typically set up a kickoff meeting where project stakeholders are gathered to discuss the project as well as the responsibilities of participants. During this meeting we verify the scope of the engagement and discuss the project activities. It's an opportunity for both sides to ask questions and get to know each other. By the end of the kickoff there is an understanding of the following:

- Designated points of contact
- Communication methods and frequency
- Shared documentation
- Code and/or any other artifacts necessary for project success
- Follow-up meeting schedule, such as a technical walkthrough
- Understanding of timeline and duration

RAMP-UP

Ramp-up consists of the activities necessary to gain proficiency on the project. This can include the steps needed for familiarity with the codebase or technological innovation utilized. This may include, but is not limited to:

- Reviewing previous work in the area including academic papers
- Reviewing programming language constructs for specific languages

- Researching common flaws and recent technological advancements

Review

The review phase is where most of the work on the engagement is completed. This is the phase where we analyze the project for flaws and issues that impact the security posture. Depending on the project this may include an analysis of the architecture, a review of the code, and a specification matching to match the architecture to the implemented code.

In this code audit, we performed the following tasks:

1. Security analysis and architecture review of the original protocol
2. Review of the code written for the project
3. Compliance of the code with the provided technical documentation

The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools were used to assist the reviewer during the testing. We discuss our methodology in more detail in the following sections.

Code Safety

We analyzed the provided code, checking for issues related to the following categories:

- General code safety and susceptibility to known issues
- Poor coding practices and unsafe behavior
- Leakage of secrets or other sensitive data through memory mismanagement
- Susceptibility to misuse and system errors
- Error management and logging

This list is general and not comprehensive, meant only to give an understanding of the issues we are looking for.

Technical Specification Matching

We analyzed the provided documentation and checked that the code matches the specification. We checked for things such as:

- Proper implementation of the documented protocol phases

- Proper error handling
- Adherence to the protocol logical description

Reporting

FYEO Inc. delivers a draft report that contains an executive summary, technical details, and observations about the project.

The executive summary contains an overview of the engagement including the number of findings as well as a statement about our general risk assessment of the project. We may conclude that the overall risk is low but depending on what was assessed we may conclude that more scrutiny of the project is needed.

We report security issues identified, as well as informational findings for improvement, categorized by the following labels:

- Critical
- High
- Medium
- Low
- Informational

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

As we perform the audit, we may identify issues that aren't security related, but are general best practices and steps that can be taken to lower the attack surface of the project. We will call those out as we encounter them and as time permits.

As an optional step, we can agree on the creation of a public report that can be shared and distributed with a larger audience.

Verify

After the preliminary findings have been delivered, this could be in the form of the approved communication channel or delivery of the draft report, we will verify any fixes within a window of time specified in the project. After the fixes have been verified, we will change the status of the finding in the report from open to remediated.

The output of this phase will be a final report with any mitigated findings noted.

Additional Note

It is important to note that, although we did our best in our analysis, no code audit or assessment is a guarantee of the absence of flaws. Our effort was constrained by resource and time limits along with the scope of the agreement.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination.

The Classification of vulnerabilities

Security vulnerabilities and areas for improvement are weighted into one of several categories using, but is not limited to, the criteria listed below:

Critical – vulnerability will lead to a loss of protected assets

- This is a vulnerability that would lead to immediate loss of protected assets
- The complexity to exploit is low
- The probability of exploit is high

High - vulnerability has potential to lead to a loss of protected assets

- All discrepancies found where there is a security claim made in the documentation that cannot be found in the code
- All mismatches from the stated and actual functionality
- Unprotected key material
- Weak encryption of keys
- Badly generated key materials
- Txn signatures not verified
- Spending of funds through logic errors
- Calculation errors overflows and underflows

Medium - vulnerability hampers the uptime of the system or can lead to other problems

- Insecure calls to third party libraries
- Use of untested or nonstandard or non-peer-reviewed crypto functions

- Program crashes, leaves core dumps or writes sensitive data to log files

Low – vulnerability has a security impact but does not directly affect the protected assets

- Overly complex functions
- Unchecked return values from 3rd party libraries that could alter the execution flow

Informational

- General recommendations