HashEye

// SMART CONTRACT SECURITY AUDIT

SEDA Chain Token Migration

Security assessment by HashEye · prepared for SEDA

HASHEYE AUDITED

PROJECT	SEDA Chain Token Migration
PERFORMED BY	HashEye Security Review Team - Sergey, Brian
RESEARCH	Sergey, Brian
CLIENT	SEDA
CATEGORY	Blockchain
PUBLISHED	March 1, 2024
REPORT ID	research-seda-chain-token-migration-2024-03-01-ekvntc

This report was produced by HashEye's security review team under the HashEye review process, combining source review, deployment-state checks, and senior manual verification. Full report detail is available at hasheye.io/audits/research-seda-chain-token-migration-2024-03-01-ekvntc.

SEDAChainTokenMigration SecurityAssessment March29,2024 Preparedfor: JasperdeGooijer SEDA Preparedby:BenjaminSamuelsandMichaelColburn

Abouthasheye Foundedin2012andheadquarteredinNewYork,hasheyeprovidestechnicalsecurity assessmentandadvisoryservicestosomeoftheworld'smosttargetedorganizations.We combinehigh-endsecurityresearchwithareal -worldattackermentalitytoreduceriskand fortifycode.With100+employeesaroundtheglobe,we'vehelpedsecurecriticalsoftware elementsthatsupportbillionsofendusers,includingKubernetesandtheLinuxkernel. Wemaintainanexhaustivelistofpublicationsathttps://github.com/hasheye-io/publications, withlinkstopapers,presentations,publicauditreports,andpodcastappearances. Inrecentyears,hasheyeconsultantshaveshowcasedcutting-edgeresearchthrough presentationsatCanSecWest,HCSS,Devcon,EmpireHacking,GrrCon,LangSec,NorthSec, theO'ReillySecurityConference,PyCon,REcon,SecurityBSides,andSummerCon. Wespecializeinsoftwaretestingandcodereviewprojects,supportingclientorganizations inthetechnology,defense,andfinanceindustries,aswellasgovernmententities.Notable clientsincludeHashiCorp,Google,Microsoft,WesternDigital,andZoom. hasheyealsooperatesacenterofexcellencewithregardtoblockchainsecurity.Notable projectsincludeauditsofAlgorand,BitcoinSV,Chainlink,Compound,Ethereum2.0, MakerDAO,Matic,Uniswap,Web3,andZcash. Tokeepuptodatewithourlatestnewsandannouncements,pleasefollowhasheyeon Twitterandexploreourpublicrepositoriesathttps://github.com/hasheye-io.Toengageus directly,visitour"Contact"pageathttps://www.hasheye.io/contact,oremailusat info@hasheye.io. hasheye,Inc. 497CarrollSt.,Space71,SeventhFloor Brooklyn,NY11215 https://www.hasheye.io info@hasheye.io hasheye 1SEDAChainTokenMigrationSecurityAssessment PUBLIC

NoticesandRemarks CopyrightandDistribution ©2024byhasheye,Inc. Allrightsreserved.hasheyeherebyassertsitsrighttobeidentifiedasthecreatorofthis reportintheUnitedKingdom. Thisreportisconsideredbyhasheyetobepublicinformation;itislicensedtoSEDA underthetermsoftheprojectstatementofworkandhasbeenmadepublicatSEDA's request.Materialwithinthisreportmaynotbereproducedordistributedinpartorin wholewithouttheexpresswrittenpermissionofhasheye. ThesolecanonicalsourceforhasheyepublicationsisthehasheyePublicationspage. Reportsaccessedthroughanysourceotherthanthatpagemayhavebeenmodifiedand shouldnotbeconsideredauthentic. TestCoverageDisclaimer Allactivitiesundertakenbyhasheyeinassociationwiththisprojectwereperformedin accordancewithastatementofworkandagreeduponprojectplan. Securityassessmentprojectsaretime-boxedandoftenreliantoninformationthatmaybe providedbyaclient,itsaffiliates,oritspartners.Asaresult,thefindingsdocumentedin thisreportshouldnotbeconsideredacomprehensivelistofsecurityissues,flaws,or defectsinthetargetsystemorcodebase. hasheyeusesautomatedtestingtechniquestorapidlytestthecontrolsandsecurity propertiesofsoftware.Thesetechniquesaugmentourmanualsecurityreviewwork,but eachhasitslimitations:forexample,atoolmaynotgeneratearandomedgecasethat violatesapropertyormaynotfullycompleteitsanalysisduringtheallottedtime.Theiruse isalsolimitedbythetimeandresourceconstraintsofaproject. hasheye 2SEDAChainTokenMigrationSecurityAssessment PUBLIC

TableofContents Abouthasheye1 NoticesandRemarks2 TableofContents3 ProjectSummary4 ExecutiveSummary5 ProjectGoals7 ProjectTargets8 ProjectCoverage9 AutomatedTesting11 CodebaseMaturityEvaluation12 SummaryofFindings14 DetailedFindings15 1.Sensitivematerialstoredinfileswithloosepermissions15 2.panic()isoverusedforerrormanagement17 A.VulnerabilityCategories19 B.CodeMaturityCategories21 C.CodeQualityRecommendations23 D.AutomatedStaticAnalysis24 E.FixReviewResults26 DetailedFixReviewResults27 F.FixReviewStatusCategories28 hasheye 3SEDAChainTokenMigrationSecurityAssessment PUBLIC

ProjectSummary ContactInformation Thefollowingprojectmanagerwasassociatedwiththisproject: SamGreenup,ProjectManager sam.greenup@hasheye.io Thefollowingengineeringdirectorwasassociatedwiththisproject: JosselinFeist,EngineeringDirector,Blockchain josselin.feist@hasheye.io Thefollowingconsultantswereassociatedwiththisproject: BenjaminSamuels,ConsultantMichaelColburn,Consultant benjamin.samuels@hasheye.iomichael.colburn@hasheye.io ProjectTimeline

Thesignificanteventsandmilestonesoftheprojectarelistedbelow. DateEvent February8,2024Technicalonboardingcall February22,2024Pre-projectkickoffcall March4,2024Statusupdatemeeting#1 March11,2024Deliveryofreportdraft March11,2024Reportreadoutmeeting March29,2024Deliveryofcomprehensivereportwithfixreviewappendix hasheye 4SEDAChainTokenMigrationSecurityAssessment PUBLIC

ExecutiveSummary EngagementOverview SEDAengagedhasheyetoreviewthesecurityofitstokenmigrationcontractsandthe SEDAchain'sstakingandvestingmodules. AteamoftwoconsultantsconductedthereviewfromFebruary26toMarch8,2024,fora totaloffourengineer-weeksofeffort.Ourtestingeffortsfocusedonissuesinthetoken migrationprocessorinthestakingorvestingoperationsintheirrespectivemodules.With fullaccesstosourcecodeanddocumentation,weperformedstaticanddynamictestingof thetargets,usingautomatedandmanualprocesses. ObservationsandImpact Thecustomlogicforthetokenmigrationisstraightforwardandpresentsaminimalattack surfaceinthesmartcontractsthemselves.Themigrationprocesswasdesignedtoinclude multiplesafeguardssuchasarelayerallowlistandtheabilitytopausethemigration contractsoneitherside. FortheSEDAchaincodebaseitself,thecurrentcustomfeaturesetisminimalandincludes supportonlyforstakingandvestingofthemigratedtoken.Ratherthanreimplementing thisfromscratch,muchofthecodehasbeenadaptedfromanexistingproject.Asidefrom themodificationsnecessarytoadaptthelogic,theSEDAteamalsotookamore comprehensiveapproachtoerrorhandling,thoughthisincludessomepanicstatements thatseemoutofplace(TOB-SEDA-2). Recommendations Basedonthecodebasematurityevaluationandfindingsidentifiedduringthesecurity review,hasheyerecommendsthatSEDAtakethefollowingsteps: • Remediatethefindingsdisclosedinthisreport.Thesefindingsshouldbe addressedaspartofadirectremediationoraspartofanyrefactorthatmayoccur whenaddressingotherrecommendations. • Continueworkonsimulationsandaddafuzztestingsuite.Whilesomework hasbeendoneforsimulationsinSEDA,werecommendcontinuingthisworktohelp establishafuzztestingsuitefortheSEDAchain.Oncesimulationsareworking,the nativeGofuzzingenginecanbeaddedtorandomlyexploretheentirechainlifecycle fromgenesistogenerationoftransactionstoblockcreation.Thismayprove especiallyusefulfortestingedgecaseswiththenon-deterministicOverlaynetwork, butsincetheseintegrationscanoftenbecomplex,planningsuchatestsuiteshould bestartedearly.Someguidelinesforestablishingsuchatestharnesscanbefound onourblog:ImprovingthestateofCosmosfuzzing. hasheye 5SEDAChainTokenMigrationSecurityAssessment PUBLIC

• ConsideraddingCosmosinvariants.AsSEDAcontinuestoimplementelementsof theOverlaynetwork,itmaybebeneficialtocreateaninvariantregistryforthechain toincreaseitslevelofsecurityassurance.Cosmosprovidessupportforruntime invariantverification,whichcanhaltthechainorperformothermitigationactionsin anemergencyifaninvariantisviolated.Thismayproveespeciallyusefulifthe Overlaycomponenthascriticallivenessconsiderationsthatcouldnegativelyimpact SEDAiftheyarebreached. FindingSeveritiesandCategories Thefollowingtablesprovidethenumberoffindingsbyseverityandcategory. EXPOSUREANALYSIS SeverityCount High0 Medium0 Low1 Informational1 Undetermined0 CATEGORYBREAKDOWN CategoryCount Configuration1 ErrorReporting1 hasheye 6SEDAChainTokenMigrationSecurityAssessment PUBLIC

ProjectGoals Theengagementwasscopedtoprovideasecurityassessmentofthe SEDAchain. Specifically,wesoughttoanswerthefollowingnon-exhaustivelistofquestions: • Canthedamagefromapotential Wormholehackbemitigatedorlimited? • Cananattackerdouble-migratetokensorreplaymigrationrequests? • Doeseachofthesystem'sprivilegedcomponents(CosmWasmcontract,Solidity migrationcontract,etc.)useadequateauthorization? • Wereanyarithmeticorlogicissuesintroducedbythevestingmodulechanges? • Isitpossibletohaltthechainbycausinga Cosmosmodulepanic? hasheye 7SEDAChainTokenMigrationSecurityAssessment PUBLIC

ProjectTargets Theengagementinvolvedareviewandtestingofthetargetslistedbelow. seda-chain Repositoryhttps://github.com/sedaprotocol/seda-chain/ Version 8a3e964098f16b37582dfa14ca0d00652c525972 TypeGo PlatformCosmos token-migration Repositoryhttps://github.com/sedaprotocol/token-migration/ Version aa108d11921802e90faa62a45afd86e710e78e9e TypeRust,Solidity PlatformCosmWasm,EVM hasheye 8SEDAChainTokenMigrationSecurityAssessment PUBLIC

ProjectCoverage Thissectionprovidesanoverviewoftheanalysiscoverageofthereview,asdeterminedby ourhigh-levelengagementgoals.Ourapproachesincludedthefollowing: • Cosmosstakingmodule.Thismoduleisforkedfromthemainline Cosmosstaking module,withchangesthatallowtokendelegationsandunbondingdelegationstobe transferredtoothervalidators.Weanalyzedthechangestothismoduleusing SAST toolingandmanuallyreviewedthemforarithmeticandlogicissues. • Cosmosvestingmodule.Thismoduleisalsoforkedfromthemainline Cosmos stakingmodule,withchangesthatallowthecreatorofavestingcontractto“claw back”vestingfunds.Thisclawbackcodewasadaptedfromcodeby Agorictoaddthe abilitytoclawbackcontinuouslyvestingfundsinsteadofonlyperiodicallyvesting funds.Weanalyzedthismoduleusing SASTtoolingandmanuallyreviewedit for issuessuchastheuseofvestingmodulestobypasslockedtokens,unauthorized creationofavestingaccountforatargetuser,andotherlogicissues. • Soliditycontract.The MigrateFLX contractallowsuserstoinitiateatoken migrationbyburningtheirtokensontheEthereumsideandsignalingthistothe Wormholecontract.Wemanuallyreviewedthiscontracttocheckwhethertokens areburnedproperlyandcanberecovered,whethertheappropriateparametersare passedtoWormhole,andwhetherthepausingfunctionalityworksasexpected. • CosmWasmcontract.The cw-seda-token-migration contractreceives messagesrelayedviatheWormholeprotocolanddeliversthemigratedtokensto theenduser.Wemanuallyreviewedthiscontracttocheckwhetherthemigration logicperformspropervalidationoftheVerifiedActionApprovals(VAAs)from Wormhole,whethertherearеanyerrorsinbookkeeping,whethertherelayer allowlistcanbebypassed,andwhetherproperaccesscontrolsareinplacefor managingthegovernanceoperations.Wealsocheckedforanyissuesrelatedto addressnormalizationorotherstandardCosmWasmissues. CoverageLimitations Becauseofthetime-boxednatureoftestingwork,itiscommontoencountercoverage limitations.Duringthisproject,wewereunabletoperformcomprehensivetestingofthe followingsystemelements,whichmaywarrantfurtherreview: • TheSEDAteamnotedthatthe randomness and wasm-storage modulesinthe seda-chain repositorywerenotusedoutsideoftests,sotheywereconsideredout ofscopeforthisreview. hasheye 9SEDAChainTokenMigrationSecurityAssessment PUBLIC

• TheWormholeSoliditycontract,theCosmWasmcontract,andthenetworkthat bridgesmessagesbetweenthetwowereoutofscope.Acompromiseofthe Wormholenetworkwouldleadtoasystemcompromisewhosedamagewouldhave tobemitigatedinatimelymannerbytheSEDAteam. • SEDA’suseofVerifiableRandomnessFunctions(VRF)wasoutofscopeforthisaudit. hasheye 10SEDAChainTokenMigrationSecurityAssessment PUBLIC

AutomatedTesting hasheyeusesautomatedtechniquestoextensivelytestthesecuritypropertiesof software.Weusebothopen-sourcestaticanalysisandfuzzingutilities,alongwithtools developedinhouse,toperformautomatedtestingofsourcecodeandcompiledsoftware. TestHarnessConfiguration Weusedthefollowingtoolsintheautomatedtestingphaseofthisproject: ToolDescriptionPolicy SlitherAstaticanalysisframeworkthatcanstaticallyverify algebraicrelationshipsbetweenSolidityvariables N/A SemgrepAnopen-sourcestaticanalysistoolforfindingbugsand enforcingcodestandardswheneditingorcommitting codeandduringbuildtime. AppendixD GosecAstaticanalysisutilitythatlooksforvariousproblemsin Gocodebasesandwillidentifypotentiallystored credentials,unhandlederrors,cryptographically troublingpackages,andsimilarproblems AppendixD Staticcheck Astaticanalysisutilitythatidentifiesbothstylistic problemsandimplementationproblemswithinaGo codebase AppendixD ineffassignAstaticanalysisutilitythatidentifiesineffectual assignments,mostnotablyineffectualerrorassignments AppendixD errcheckAstaticanalysisutilitythatidentifiessituationswhere errorsarenothandledappropriately AppendixD looppointerAnanalyzerthatchecksforpointerstoenclosingloop variables AppendixD NilAwayAstaticanalysistoolfordetectingnilpanics,whichcan beespeciallydetrimentalinaCosmosapplication AppendixD hasheye 11SEDAChainTokenMigrationSecurityAssessment PUBLIC

CodebaseMaturityEvaluation hasheyeusesatraffic-lightprotocoltoprovideeachclientwithaclearunderstandingof theareasinwhichitscodebaseismature,immature,orunderdeveloped.Deficiencies identifiedhereoftenstemfromrootcauseswithinthesoftwaredevelopmentlifecyclethat shouldbeaddressedthroughstandardizationmeasures(e.g.,theuseofcommonlibraries, functions,orframeworks)ortrainingandawarenessprograms. CategorySummaryResult ArithmeticTheSEDAchainhasrelativelylittlearithmeticinits modifiedcode,withmostarithmeticrisklimitedto operationsimplementedinthird-partycomponents

upstreamofSEDA’schanges. Not Applicable AuditingTheCosmWasmandSolidityContractsEmitEventsWhere StateMutationsAreCreated.TheCosmosmodulesall emitPrometheusmetricsforimportantmessagesand statemutations. Satisfactory Authentication/ AccessControls EachMigrationContractHasAdequateAccessControls implementedforprivilegedfunctions,limitingaccesstoa specificownerorgovernanceaddress.TheCosmWasm migrationcontracthasadditionalauthorizationforthe forwardingofVAAmessagesfromtheWormhole network,creatingadefense-in-depthmeasurethatcan protectSEDA’stokenmigrationintheeventofa Wormholecompromise. Strong Complexity Management SEDAreusespreexistingcomponentsforitsvesting, staking,andclawbackfunctionality,reducingtheamount ofcodetheteamhastomaintainandsimplifying maintenance.SEDAalsohasexcellentinternal documentationandspecification,boostingthesystem’s maintainability. However,somefunctionsfromtheAgoriccodebaseare excessivelylargeanddifficulttounderstand,withsome running200-pluslinesofcode.Thesefunctionseither needtoberefactoredorhavefurtherdocumentation addedtoexplainwhattheyaredoingandtheirside effects. Satisfactory hasheye 12SEDAChainTokenMigrationSecurityAssessment PUBLIC

DecentralizationSEDA’scentralizationrisksareadequatelyhighlightedin itssupportingdocumentation,anddefense-in-depth measureshavebeenaddedforaddressingcentralization risksintroducedbytheWormholeintegration. Satisfactory DocumentationThetokenmigrationprocessisextensivelydocumented. Thespecificationoutlineseachofthekeycomponentsin thesystem,theirfeaturesandrequirements,details abouttheirimplementation,andwhichpartyis responsiblefortheiroperation. Strong Low-Level Manipulation Thiscategorywasnotapplicableforthisreview.Not Applicable MemorySafety andError Handling Forthemostpart,thecodebasetakesavery comprehensiveanddefensiveapproachtoerror handling.However,thereareafewinstancesinthe Cosmosmoduleswherepanicsunexpectedlyoccur insteadofanerrorbeingreturned(TOB-SEDA-2). Satisfactory Testingand Verification Overallthecodebasehasgoodtestcoverage.The Cosmoscodewouldbenefitfromtheadditionofmore advancedtestingtechniques,suchastestinginvariants andfuzzing.Additionally,thesimulationsshouldbe completed. Satisfactory Transaction Ordering Withinthecurrentscopeofthesystem’scapabilities, therearenovectorsfortransactionorderingrisks.This maychangeasfunctionalitybecomesavailableinfuture versions. Strong hasheye 13SEDAChainTokenMigrationSecurityAssessment PUBLIC

SummaryofFindings Thetablebelowsummarizesthefindingsofthereview,includingtypeandseveritydetails. IDTitleTypeSeverity 1Sensitivematerialstoredinfileswithloose permissions ConfigurationLow 2panic()isoverusedforerrormanagementErrorReportingInformational hasheye 14SEDAChainTokenMigrationSecurityAssessment PUBLIC

DetailedFindings 1.Sensitivematerialstoredinfileswithloosepermissions Severity:LowDifficulty:High Type:ConfigurationFindingID:TOB-SEDA-1 Target: seda-chain/app/utils/vrf_key.go , seda-chain/cmd/sedad/cmd/init.go , seda-chain/e2e/validator.go Description TheSEDAnodeclientstoressensitivematerialusingoverlyloosefileorfolderpermissions of 755 orhigher.ThisincludestheVRFkeyfileinfigures1.1and1.2,thevalidatorstatefile infigure1.3,andthevalidatorconfigurationfileinfigure1.4.Thefinalexampleisinafile thatinstrumentsend-to-endtestingandisincludedforthoroughness.

```
pvKeyFile:=config.PrivValidatorKeyFile()
iferr:=os.MkdirAll(filepath.Dir(pvKeyFile),0o755);err≠nil{
returnnil,fmt.Errorf("couldnotcreatedirectory%q:%w", filepath.Dir(pvKeyFile),err) }
```

Figure1.1:Thevalidatorkeyfileisstoredinadirectorywithpermissioncode 755 ,allowingevery useronthesystemtoviewthefile.(seda-chain/app/utils/vrf_key.go#229-232)

```
keyFilePath:=config.PrivValidatorKeyFile()
iferr:=os.MkdirAll(filepath.Dir(keyFilePath),0o777);err≠nil{
returnfmt.Errorf("couldnotcreatedirectory%q:%w", filepath.Dir(keyFilePath),err) }
```

Figure1.2:Thevalidatorkeyfileisstoredinadirectorywithpermissioncode 777 , allowingeveryuseronthesystemtoviewandmodifythefile. (seda-chain/cmd/sedad/cmd/init.go#40-43)

```
stateFilePath:=config.PrivValidatorStateFile()
iferr:=os.MkdirAll(filepath.Dir(stateFilePath),0o777);err≠nil{
returnfmt.Errorf("couldnotcreatedirectory%q:%w", filepath.Dir(stateFilePath),err) }
```

Figure1.3:Thevalidatorstatefileisstoredinadirectorywithpermissioncode 777 , allowingeveryuseronthesystemtoviewandmodifythefile. (seda-chain/cmd/sedad/cmd/init.go#44-47)

hasheye 15SEDAChainTokenMigrationSecurityAssessment PUBLIC

```
func(v*validator)createConfig()error{ p:=path.Join(v.configDir(),"config")
returnos.MkdirAll(p,0o755) }
```

Figure1.4:Thenodeconfigurationfileisstoredinadirectorywithpermissioncode 755 ,allowing everyuseronthesystemtoviewthefile.(seda-chain/e2e/validator.go#64-67)

Thecodepathinfigure1.4ispartofthetestsuite,notproductioncode. ExploitScenario AvalidatoroperatorrunsanodeonamajorcloudplatformthatincludesalOw-permission

loggingandmetricsdaemononeachcomputeinstanceusedbyitsclients.SinceSEDA’skey materialisstoredinfilespermissionedsothatanyuseronthesystemmayviewtheir contents,compromiseoftheloggingdaemonleadstocompromiseoftheprivatekey. Recommendations Shortterm,modifythecreateddirectorypermissionstousepermissioncode 0750 ,ora morerestrictivecodeifpossible.Thismodificationshouldincludelesssensitivefilessuch astheconfigurationandstatefilesbecauseenforcingrulesaroundfilepermissionsis easierifthe rulesarethesameregardlessoffilecontent. Longterm,establishinternalcodequalityguidelinesthatestablishthepropermethods throughwhichfilescanbecreated.ThisfindingwasdiscoveredusingGosec,sotheSEDA repositorymaybenefitfrommoreregularuseofGosecorfrommakingitarequirementin therepository’sCIpipeline. hasheye 16SEDAChainTokenMigrationSecurityAssessment PUBLIC

2.panic()isoverusedforerrormanagement Severity:InformationalDifficulty:N/A Type:ErrorReportingFindingID:TOB-SEDA-2 Target:Stakingmodule,vestingmodule Description TheSEDACosmosmodulesforstakingandvestinguse panic() statementsforcertain errorcases,asshowninfigures2.1and2.2.Go’spanicmechanismisnotrecommendedfor useinGoapplicationsexceptforthemostfatalerrorsthatrequireanimmediatehaltof theapplication’sruntime. maxEntries,err:=k.MaxEntries(ctx) iferr≠nil{ panic(err) } valSrcAddr,err:=sdk.ValAddressFromBech32(toRedelegation.ValidatorSrcAddress) iferr≠nil{ panic(err) } valDstAddr,err:=sdk.ValAddressFromBech32(toRedelegation.ValidatorDstAddress) iferr≠nil{ panic(err) } Figure2.1:The panic() statementisusedtoimmediatelyterminatetheapplicationinthe stakingmodule.(seda-chain/x/staking/keeper/keeper.go#62-74) if!toClawBackStaking.IsZero(){ panic("failedtoclawbackfullamount") } Figure2.2:The panic() statementisusedtoimmediatelyterminatetheapplicationinthe vestingmodule.(seda-chain/x/vesting/keeper/msg_server.go#257-259) ExploitScenario Thecodecontaining panic() statementsiscalledbyout-of-blockcode(e.g.,duringthe BeginBlock and EndBlock functions),andthepaniccausesthechaintohalt. Inthechain’scurrentstate,thisscenarioisnotpossible,butgivenSEDA’splannedchanges, consideravoidingthesepanic-inducedchainhalts. hasheye 17SEDAChainTokenMigrationSecurityAssessment PUBLIC

Recommendations Shortterm,changeusesof panic() tonormalGoerrorswhereapplicable. Longterm,considerbanningtheuseof panic() inSEDA’scode-contributingguidelines. hasheye 18SEDAChainTokenMigrationSecurityAssessment PUBLIC

A.VulnerabilityCategories Thefollowingtablesdescribethevulnerabilitycategories,severitylevels,anddifficulty levelsusedinthisdocument. VulnerabilityCategories CategoryDescription AccessControlsInsufficientauthorizationorassessmentofrights AuditingandLoggingInsufficientauditingofactionsorloggingofproblems AuthenticationImproperidentificationofusers ConfigurationMisconfiguredservers,devices,orsoftwarecomponents CryptographyAbreachofsystemconfidentialityorintegrity DataExposureExposureofsensitiveinformation DataValidationImproperrelianceonthestructureorvaluesofdata DenialofServiceAsystemfailurewithanavailabilityimpact ErrorReportingInsecureorinsufficientreportingoferrorconditions PatchingUseofanoutdatedsoftwarepackageorlibrary SessionManagementImproperidentificationofauthenticatedusers TestingInsufficienttestmethodologyortestcoverage TimingRaceconditionsorotherorder-of-operationsflaws UndefinedBehaviorUndefinedbehaviortriggeredwithinthesystem hasheye 19SEDAChainTokenMigrationSecurityAssessment PUBLIC

SeverityLevels SeverityDescription InformationalTheissuedoesnotposeanimmediateriskbutisrelevanttosecuritybest practices. UndeterminedTheextentoftheriskwasnotdeterminedduringthisengagement. LowTheriskissmallorisnotonetheclienthasindicatedisimportant. MediumUserinformationisatrisk;exploitationcouldposereputational,legal,or moderatefinancialrisks. HighTheflawcouldaffectnumeroususersandhaveseriousreputational,legal, orfinancialimplications. DifficultyLevels DifficultyDescription UndeterminedThedifficultyofexploitationwasnotdeterminedduringthisengagement. LowTheflawiswellknown;publictoolsforitsexploitationexistorcanbe scripted. MediumAnattackermustwriteanexploitorwillneedin-depthknowledgeofthe system. HighAnattackermusthaveprivilegedaccesstothesystem,mayneedtoknow complextechnicaldetails,ormustdiscoverotherweaknessestoexploitthis issue. hasheye 20SEDAChainTokenMigrationSecurityAssessment PUBLIC

B. Code Maturity Categories

The following tables describe the code maturity categories and rating criteria used in this document.

Code Maturity Categories	Category Description
Arithmetic	The proper use of mathematical operations and semantics
Auditing	The use of event auditing and logging to support monitoring
Authentication/ Access Controls	The use of robust access controls to handle identification and authorization and to ensure safe interactions with the system
Complexity Management	The presence of clear structures designed to manage system complexity, including the separation of system logic into clearly defined functions
Cryptography and Key Management	The safe use of cryptographic primitives and functions, along with the presence of robust mechanisms for key generation and distribution
Decentralization	The presence of a decentralized governance structure for mitigating insider threats and managing risks posed by contract upgrades
Documentation	The presence of comprehensive and readable codebase documentation
Low-Level Manipulation	The justified use of inline assembly and low-level calls
Memory Safety and Error Handling	The presence of memory safety and robust error-handling mechanisms
Testing and Verification	The presence of robust testing procedures (e.g., unit tests, integration tests, and verification methods) and sufficient test coverage
Transaction Ordering	The system's resistance to transaction-ordering attacks

Rating	Criteria
Strong	No issues were found, and the system exceeds industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.
Weak	Many issues that affect system safety were found.
Missing	A required component is missing, significantly affecting system safety.
Not Applicable	The category is not applicable to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

C. Code Quality Recommendations

The following recommendations are not associated with specific vulnerabilities. However, they can enhance code readability and may prevent the introduction of vulnerabilities in the future.

- The seda-chain codebase contains several "TODO" or "TO-DO" notes. Ideally these should be in an actual issue tracker instead of left in the code itself. This would allow additional context to be included that can explain what exactly needs to be done, as well as what was preventing it from being implemented at the time.
- Split the CosmWasm contract's `execute_transfer_gov` function into two separate functions: `execute_propose_gov` and `execute_accept_gov`. This pattern, similar to what is done in the Ownable2Step contract from OpenZeppelin's Solidity library, could help prevent mistakenly transferring governance of the contract to the wrong address.

D. Automated Static Analysis

This appendix describes the setup of the automated analysis tools used during this audit. Though static analysis tools frequently report false positives, they detect certain categories of issues, such as memory leaks, misspecified format strings, and the use of unsafe APIs, with essentially perfect precision. We recommend periodically running these static analysis tools and reviewing their findings. Some static analysis tools provide a high enough signal to be worth integrating into the project's CI/CD pipeline; however, the ideal set of tools tends to vary from project to project.

Semgrep To install Semgrep, we used pip by running `python3 -m pip install semgrep`. We used version 1.64.0 of Semgrep. To run Semgrep on the codebase, we ran the following in the root directory of the project: `semgrep --config "p/hasheye" --sarif --metrics=off --output semgrep.sarif`. We also ran the tool with the following rules (configurations):

- p/ci
- p/security-audit
- r/go.lang
- p/semgrep-go-correctness

We recommend integrating Semgrep into the project's CI/CD pipeline. Integrate at least the rules with HIGH confidence and with MEDIUM confidence and HIGH impact. In addition to the four configurations listed above, we recommend using hasheye's set of Semgrep rules (from the repository or less preferably from the registry).

Other Static Analysis Tools We recommend using the following tools, either in an ad hoc manner or by integrating them into the CI/CD pipeline:

- golangci-lint : This tool is a wrapper around various other tools (some but not all of which are listed below). It offers the most effective path to integrating some of the following tools into your CI/CD pipeline because it provides centralized SAST configuration.

• Gosecisastaticanalysisutilitythatlooksforvariousproblemsin Gocodebases. Notably, Gosecwillidentifypotentiallystoredcredentials, unhandlederrors, cryptographicallytroublingpackages, andsimilarproblems. • Go-vet isaverypopularstaticanalysisutilitythatsearchesformoreGo-specific problemswithinacodebase, suchasmistakespertainingtoclosures, marshaling, andunsafepointers. Go-vet isintegratedwithinthe go commanditself, with supportforothertoolsthroughthe vettool commandlineflag. • Staticcheckisastaticanalysisutilitythatidentifiesbothstylisticproblemsand implementationproblemswithinaGocodebase. Manyofthestylisticproblems Staticcheckidentifiesarealsoindicativeofpotentialproblemareasinaproject. • ineffassignisastaticanalysisutilitythatidentifiesineffectualassignments. These ineffectualassignmentsoftenidentifysituationswhereerrorsgounchecked, which couldleadtoundefinedbehavioroftheprogramduetoexecutioninaninvalid programstate. • errcheckisastaticanalysisutilitythatidentifiessituationswhereerrorsarenot handledappropriately. • looppointerisananalyzerthatchecksforpointerstoenclosingloopvariables. • exportlooprefisanotheranalyzerthatchecksforpointerstoenclosingloop variableswithmoreaccurateresultsthanlooppointerbutproducesfalse-negatives forcertaincases. • NilAwayisastaticanalysistoolfordetectingnilpanics. PleasealsorefertoourblogpostonSecurityassessmenttechniquesforGoprojectsfor furtherdiscussionoftheGo-relatedanalysistools.

E. FixReviewResults Whenundertakingafixreview, hasheyereviewsthefixesimplementedforissues identifiedintheoriginalreport. Thisworkinvolvesareviewofspecificareasofthesource codeandsystemconfiguration, notcomprehensiveanalysisofthesystem. OnMarch21, 2024, hasheyereviewedthefixesandmitigationsimplementedbythe SEDAteamfortheissuesidentifiedinthisreport. Wereviewedeachfixtodetermineits effectivenessinresolvingtheassociatedissue. Insummary, ofthetwoissuesdescribedinthisreport, SEDAhasresolvedbothofthem. For additionalinformation, pleaseseetheDetailedFixReviewResultsbelow. TheSEDAteam alsoaddressedthecodequalityrecommendationsinappendixC. IDTitleStatus 1Sensitivematerialstoredinfileswithloosepermissions Resolved 2panic()isoverusedforerrormanagement Resolved

DetailedFixReviewResults TOB-SEDA-1: Sensitivematerialstoredinfileswithloosepermissions ResolvedinPR#212. Thedirectoriesarenowbeingcreatedwithamorerestrictive permissioncode, 0700 . TOB-SEDA-2: panic()isoverusedforerrormanagement ResolvedinPR#212. Thehighlightedpanicshavebeenreplacedwithmoredescriptive errormessages.

F. FixReviewStatusCategories Thefollowingtabledescribesthestatusesusedtoindicatewhetheranissuehasbeen sufficientlyaddressed. FixStatus StatusDescription UndeterminedThestatusoftheissuewasnotdeterminedduringthisengagement. UnresolvedTheissuepersistsandhasnotbeenresolved. PartiallyResolvedTheissuepersistsbuthasbeenpartiallyresolved. ResolvedTheissuehasbeensufficientlyresolved.