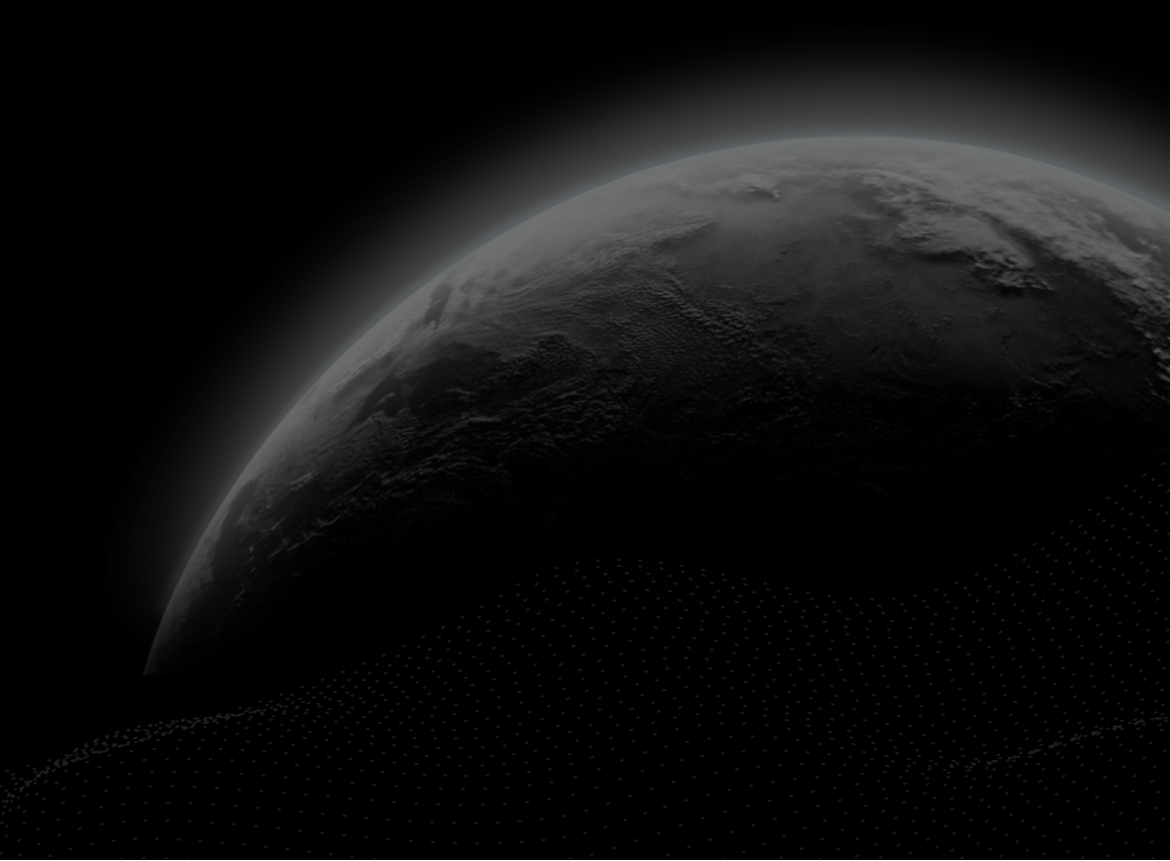
CERTIK

Security Assessment

PRYZM

CertiK Assessed on Jul 1st, 2024



CertiK Assessed on Jul 1st, 2024

PRYZM

The security assessment was prepared by CertiK, the leader in Web3.0 security.

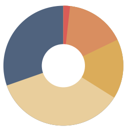
Executive Summary

TYPES	ECOSYSTEM	METHODS
Chain, DeFi	Cosmos (ATOM)	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Golang	Delivered on 07/01/2024	N/A

CODEBASE
e4ec75671e829702236a1c70a1773e7d7d3d0c15
930876154d4296a366ba2ca179c227c6663cc55b
1971f429ccc868ebf3a0a428c2d6bfb7fc7b1ec6
View All in Codebase Page

COMMITS
e4ec75671e829702236a1c70a1773e7d7d3d0c15
930876154d4296a366ba2ca179c227c6663cc55b
1971f429ccc868ebf3a0a428c2d6bfb7fc7b1ec6
View All in Codebase Page

Vulnerability Summary



56	50	0	0	6	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

🟥	1	Critical	1 Resolved	Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
🟧	9	Major	8 Resolved, 1 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
🟨	9	Medium	9 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
🟨	20	Minor	17 Resolved, 3 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
🟦	17	Informational	15 Resolved, 2 Acknowledged	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | PRYZM

CODEBASE | PRYZM

Repository

e4ec75671e829702236a1c70a1773e7d7d3d0c15 930876154d4296a366ba2ca179c227c6663cc55b

1971f429ccc868ebf3a0a428c2d6bfb7fc7b1ec6 374cad8c0f54b4f98efb6248cf434524bf2b7f65

Commit

e4ec75671e829702236a1c70a1773e7d7d3d0c15 930876154d4296a366ba2ca179c227c6663cc55b

1971f429ccc868ebf3a0a428c2d6bfb7fc7b1ec6 374cad8c0f54b4f98efb6248cf434524bf2b7f65

AUDIT SCOPE | PRYZM

376 files audited ● 3 files with Acknowledged findings ● 2 files with Resolved findings ● 371 files without findings

ID	Repo	File	SHA256 Checksum
● CLA	refractedlabs/flowtrade	x/flowtrade/keeper/claim.go	6eb0278fd4d8ac67b79f0aea2055004455 22645b49ca3371c52a195f9733d6d6
● POI	refractedlabs/flowtrade	x/flowtrade/types/position.go	d6ad56110752ee6f19061e0359a52af558 10f05c8eb5157f998578f8838ba31a
● ORD	pryzm-finance/pryzm-core	x/amm/keeper/order.go	537a3029fcbedb852a8f8581c34802447d 0369605bcb209cfff1eb8c3a3c9ef0
● FLO	refractedlabs/flowtrade	x/flowtrade/keeper/flow.go	800470134d6347da1540dbffce5359116b a7eeb5b1f2cffdc24882ce594f51b4
● POS	refractedlabs/flowtrade	x/flowtrade/keeper/position.go	940455c702606de63f59ce2d0e21beecb1 7fc8637adc0341f977d50e822b6760
● ABI	refractedlabs/flowtrade	x/flowtrade/keeper/abci.go	2bc4d630fca8b82badcb1aeff7a19b68299 ae93dc3df860afb8987f549ad6dd8
● FEE	refractedlabs/flowtrade	x/flowtrade/keeper/fee_collector.go	6ced3e5d539ed24d67c55764ed5a6199e 832b6cd468d737a76d7606e9737a8ae
● GEI	refractedlabs/flowtrade	x/flowtrade/keeper/genesis.go	dd34ac5f7d62170aa67eccec222ed6d515 646b515463931b9462e7b8504df7a1
● KEK	refractedlabs/flowtrade	x/flowtrade/keeper/keeper.go	2e92b3cfd346804d9f54516649705aa06c 6bb0f574cda30e4ba5a79ecec2afda
● MSK	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server.go	815dac4dddf1892cd926329a0d599b315 078b11a1d080fd4929285a9defd400c
● MSC	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_claim_token_in.go	ab784481234dab8b691fdeda883385fe84 88f1bd40a3877e3afa80a7f4b93941
● MSL	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_claim_token_out.go	442671c9ca42a2a7190a8f7aae29ec9f59 8fe2f3040aab78d72302d49b833420
● MSA	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_create_flow.go	834c0d2c23de289ac522f6e076fd081b04 5c0f6d5eccacbc7c960d1d2e7dfb79
● MSX	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_exit_flow.go	84e8a52f9c09dc7a48a76bf2efc04dbf7c7 0375cf39c85ecc601a7277c18f381

ID	Repo	File	SHA256 Checksum
● MSJ	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_join_flow.go	69a81c65f06055827ba057aad70aacb0a7b49c2cf668f165e8337889c7559721
● MSO	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_set_operator.go	b69dc80c571c5bf4069aac9964455b69c31f6e29081d48dcf3a857b6cfc0a484
● MSF	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_stop_flow.go	d419f8eb55111907a1789d9df5e31dea889e19745114c1d7a41b201391973a70
● MSM	refractedlabs/flowtrade	x/flowtrade/keeper/msg_server_update_params.go	88b4e1aca1f325d7f3416e4adc985be357167c948df939e0f32e8c638a702a28
● PAS	refractedlabs/flowtrade	x/flowtrade/keeper/params.go	0f8a17106dfe29568bc45b0d55ee0803c6b3773fc1178a35dc035b0627288c72
● QUY	refractedlabs/flowtrade	x/flowtrade/keeper/query.go	1da9ff9f864ae3391e0abf30a15b72ef87b729bc5a6dd30d65295ee8aceeb15f
● QUF	refractedlabs/flowtrade	x/flowtrade/keeper/query_flow.go	1f60989b62c9264df547358c39acab888e24386f1b48bcb7a91dd82986bb220f
● QUP	refractedlabs/flowtrade	x/flowtrade/keeper/query_params.go	a85935414935e83f627de6e46471b2cb692ebbebe63039e255c51834a390f257
● QUO	refractedlabs/flowtrade	x/flowtrade/keeper/query_position.go	7ebd815ee57e0b5bfba622116b2fe8f4107c7e6c10460460db7c37bf6af03cfe
● COE	refractedlabs/flowtrade	x/flowtrade/types/codec.go	3c4081129527dbbac7a23308a021985750c7cf3702a48b4777d4317a89534c08
● ERO	refractedlabs/flowtrade	x/flowtrade/types/errors.go	7c07a65cff0c11bd38fe00666cb58f23edb9cc2ceac0d8728ee451e37adb3d11
● EXT	refractedlabs/flowtrade	x/flowtrade/types/expected_keepers.go	64052d816b47d70feb9f1e33a820db623e424cbbfc33c101c29d092cfaf4d5d2
● FLW	refractedlabs/flowtrade	x/flowtrade/types/flow.go	3d3c4b43bb13e5af6565e7128605365f0625a647c5f1cce3f12425328ee186da
● GET	refractedlabs/flowtrade	x/flowtrade/types/genesis.go	013afe064abf6f918cc3f93872acbab29650d8695f58ef49e45eaeb3291d3606
● KEF	refractedlabs/flowtrade	x/flowtrade/types/key_flow.go	290d1e58d37ffd695e202a96f7999b993b5219abeb5a88ca85fc4159ad137c3a
● KEO	refractedlabs/flowtrade	x/flowtrade/types/key_position.go	1da0e9fcd8e4800aefd4f3316d5e2215ead5248eae76f53ab5a76cfedf95dff6

ID	Repo	File	SHA256 Checksum
KET	refractedlabs/flowtrade	x/flowtrade/types/keys.go	e08fc68be8eedb5c396ca047b6c24dae5fbd1cd13d711570ce4895bf053b6617
MEC	refractedlabs/flowtrade	x/flowtrade/types/message_claim_token_in.go	455c506bbefda9f3c22b2b8116e3af38c57c1479a3ea6df06610fc1fc03ccf60
MEL	refractedlabs/flowtrade	x/flowtrade/types/message_claim_token_out.go	b49812165d4736d9ba68d6167d9047cadab4c66b33f3be5f136cb9d8d43bd48f
MEF	refractedlabs/flowtrade	x/flowtrade/types/message_create_flow.go	3795d857134a778e81dce03664ac632f9781188a316e728d9d2d74e806b32033
MEX	refractedlabs/flowtrade	x/flowtrade/types/message_exit_flow.go	e90a41c1d22d5a77b4bc63252c20042b7260c05610126c88e9764ae295e03c72
MEJ	refractedlabs/flowtrade	x/flowtrade/types/message_join_flow.go	5de705230bcdb52f4d3baf3ac1db24bee0d9eb71452bee6aaf613b4e4ad6025a
MEO	refractedlabs/flowtrade	x/flowtrade/types/message_set_operator.go	cd2c14f71de166fd7946a9cde5304ddc6c8792cfc74e9be1c82772bbd767b3e6
MEW	refractedlabs/flowtrade	x/flowtrade/types/message_stop_flow.go	9e2dd6d48f6d185278ba4e8011861e5fff72aab58dce7e5136ab990e3de408b0
MEM	refractedlabs/flowtrade	x/flowtrade/types/message_update_params.go	4dbec3faed13f0b6fc046d4af020e5438cbde612ba0edc0f12a1eb16700adc9b
PAT	refractedlabs/flowtrade	x/flowtrade/types/params.go	b277c03fe6ce6875ada90e057acb1e3d544ee14b05065e5e3fa1ab1f5bbd8fb6
TYE	refractedlabs/flowtrade	x/flowtrade/types/types.go	7c347886dbeed39a02f9f23d860ffb46fa1da70151c2268a6289325c55acf415
MOE	refractedlabs/flowtrade	x/flowtrade/module.go	739658ebc5fdfe54fe3c4cf6f7fc1928dcbf576b050f5dc97220eeaa1fe137b0
ABP	refractedlabs/oracle	x/oracle/keeper/abci.go	9730c8aba4881a614c1f3ab0390368e30348a1ff1a0433f01cfbb756c137b5ec
BAL	refractedlabs/oracle	x/oracle/keeper/ballot.go	5893b53af3a4b9947c8eb8d0cbfe1b66090410e5dc52af02f6a0f40f9b0e5af7
CAL	refractedlabs/oracle	x/oracle/keeper/callbacks.go	a457bc0dd37cb22ca4072d352ebc886a3a1f7e6294d170111971ddff8d36699c
FED	refractedlabs/oracle	x/oracle/keeper/feeder_delegation.go	6fad95bea1b892af8f87cd344518c54d223fbcebedebff611427300f93e1cca0

ID	Repo	File	SHA256 Checksum
GEO	refractedlabs/oracle	x/oracle/keeper/genesis.go	b93d146709e7d2ea0d45893d84329a14e497102f79dc417d5ae8c0760f896df9
GRL	refractedlabs/oracle	x/oracle/keeper/grpc_query.go	c7dd3e27628a7e2086c05a01b9597fb754427ca684bec508e14430756c70a708
GRF	refractedlabs/oracle	x/oracle/keeper/grpc_query_feeder_delegation.go	f60900bc260712206bd88ab881217e5878639bd94065fa8ae76c825fd3163031
GRT	refractedlabs/oracle	x/oracle/keeper/grpc_query_miss_counter.go	eeeebe4ab49d6d9b9718ede15e4a9ee09d4441889c9da0b09c805e3ef00e0f7a
GRV	refractedlabs/oracle	x/oracle/keeper/grpc_query_oracle_pre_vote.go	0e7b577637b7ae65218d5fb55ebad622819120a13e92afcc6742665f0151fcb5
GRX	refractedlabs/oracle	x/oracle/keeper/grpc_query_oracle_vote.go	363c8f6bfdba8360725c0a7fe26e7bd5eb1d3fa859f4e50b68382fd94840395e
GR4	refractedlabs/oracle	x/oracle/keeper/grpc_query_params.go	599e655e3fc0ed3cf2069cfd5cfe323b6d93e6dc398ead27c60d03e0b4065db7
KE5	refractedlabs/oracle	x/oracle/keeper/keeper.go	608df0790b3b52d3151a94e3eddd0c7299e47221addebdad90dbe649308e214e
MIC	refractedlabs/oracle	x/oracle/keeper/miss_counter.go	1307c812d0ee52ffba92f8991dce9ac1c8b7acb8a5d52e295ee60400b743178a
MS6	refractedlabs/oracle	x/oracle/keeper/msg_server.go	7132025645a73d9cc7e2e50b3eff59220347a967e10f8406fe7b4eeda51569d4
MS1	refractedlabs/oracle	x/oracle/keeper/msg_server_delegate_feed_consent.go	f0a2f031ecb01c7182b98a48afceb426d53cad4d1f119b98a5365f0eb5e77cc0
MS2	refractedlabs/oracle	x/oracle/keeper/msg_server_oracle_combined_vote.go	0ef17a89c7c89df220ab9c8058a67fc2d1980b8f56e5edaefd7e666a6db190db
MSH	refractedlabs/oracle	x/oracle/keeper/msg_server_oracle_pre_vote.go	73ce51015286890811c2db52788a9047c20c8803cfe903468836f0e48cebb411
MGS	refractedlabs/oracle	x/oracle/keeper/msg_server_oracle_vote.go	c245eb17db19e15beefb8e9bbc5ad93298f2d92ea50e5be21c0843c4950e3cae
MGE	refractedlabs/oracle	x/oracle/keeper/msg_server_update_params.go	738192bbb4382d863d47f8e8a60aa6ee67c5c594af66b64fce8c72b3039cde14

ID	Repo	File	SHA256 Checksum
● ORL	refractedlabs/oracle	x/oracle/keeper/oracle_pre_vote.go	65510d7c946b8552dfa1c82e1930d07ff2038a5f1198bc08355157311178eb3c
● ORE	refractedlabs/oracle	x/oracle/keeper/oracle_vote.go	8ab5d4a19509c582e5e977c2c74986dc48cec86cb19b0f1ca84bf73d7c14cf5c
● PAO	refractedlabs/oracle	x/oracle/keeper/params.go	92e46d246bac778edb6e672498d41054dca71cd19699378c7d5e6bf723646ccd
● REW	refractedlabs/oracle	x/oracle/keeper/reward.go	ad63b97aabd76c95c827aab7e2a0f5df73b88e0c55723b045cb870157ff8f857
● SLA	refractedlabs/oracle	x/oracle/keeper/slash.go	9242475b6b37e07ea752b2851622681f1c5a56f9ff029c600fe688edcb2cab48
● TAL	refractedlabs/oracle	x/oracle/keeper/tally.go	802e986443a91e3fb0a379e729b10bb328b78653d62fe7edd811720de1931402
● VOT	refractedlabs/oracle	x/oracle/keeper/vote.go	940c7a7ce5db2a79d4a96733d4eee6b4ea18da1daa59115032c22ec031187cce
● VOI	refractedlabs/oracle	x/oracle/keeper/voting_period_time.go	bf2ddf106975b1270b5dbd523f0e277ada05c40f3ad69a5be4d551d64dd7c52d
● MOR	refractedlabs/oracle	x/oracle/module.go	45f95a533bb3e45b5af23750b431af5c27fe12d19f0fbf2ed9ebcaea9321c6b8
● TYO	refractedlabs/oracle	x/oracle/types/types.go	7c347886dbeed39a02f9f23d860ffb46fa1da70151c2268a6289325c55acf415
● PAC	refractedlabs/oracle	x/oracle/types/params.go	4a4f0708fee19412829f4a61ed45a93f5dd96af76e5ba81598e5465b1def10e3
● ME5	refractedlabs/oracle	x/oracle/types/message_update_params.go	ca2bff9bf149fb5b99d9c1394988dcd96fc6d5bbc14089269c3f14d4de5dfc91
● ME6	refractedlabs/oracle	x/oracle/types/message_oracle_vote.go	02dd86989d90a607f10572a4b0fe7d8ab3be29c419d4fd32207dfb89705422c6
● ME1	refractedlabs/oracle	x/oracle/types/message_oracle_pre_vote.go	fe6b3cb0e6daa89a3493fe1afc632953ba617c6b19631502a1f923d8ca9dc69b
● ME2	refractedlabs/oracle	x/oracle/types/message_oracle_combined_vote.go	1aefdfa4d3987de94a7c1eb7f5c63ff773d8ec465457ef057306194cb2081276
● ME9	refractedlabs/oracle	x/oracle/types/message_delegate_feed_consent.go	110df189dd108488f0a255e8e67f8be6982dbcd2c6026378ee486d2bd783ca49

ID	Repo	File	SHA256 Checksum
KE6	refractedlabs/oracle	x/oracle/types/keys.go	ffb17a9c6eba9f34c01521d7098de8bb7689a34895311234b8d3a2dd996300c0
KE1	refractedlabs/oracle	x/oracle/types/key_oracle_vote.go	d80cfc2066f1e0cead8018a5dd5e80ab13b471f6289fdeac0df4ea2a812e5ebc
KE8	refractedlabs/oracle	x/oracle/types/key_oracle_pre_vote.go	99b53a228e69e248362cac9f01096ad228a233f222fa882aada62b75c9b1b32e
KE2	refractedlabs/oracle	x/oracle/types/key_miss_counter.go	474f8fd1890c75b0eac608fea486c42878960726315ecb0faf882012a8df6775
KEG	refractedlabs/oracle	x/oracle/types/key_feeder_delegation.go	0cee6223661659ce38c7dda14022af1049f33290b19c37695a7d7ee0a3af1f14
HAS	refractedlabs/oracle	x/oracle/types/hash.go	6cdc92fa0aa9029f68563d0e863a72e4abcd059fc4784fd35ab4da2e3da5bfcb
GEL	refractedlabs/oracle	x/oracle/types/genesis.go	a2a7325be5423e650d7f0a751d623724838953715e2eac7b6e23d36233a67c91
EXS	refractedlabs/oracle	x/oracle/types/expected_keepers.go	99336170c9fde5d38a62ad19485e77d6ac0ccaa5dfbbed67b41cf745931a017c
ERP	refractedlabs/oracle	x/oracle/types/errors.go	b4cf4ca83a9144ba7c8a5dc94cf7dd355e4852044f12982f34ef845d7ef8cc6d
COP	refractedlabs/oracle	x/oracle/types/codec.go	7aea64dbaa4abb8785f8a9f0bbc9c6920a511b5860896c7acda5e50c7b2383cd
BAO	refractedlabs/oracle	x/oracle/types/ballot.go	3ce17e906867e0b65f397a45fba40f7d296930f8f15643ee947d036cb9d2e29e
ORV	refractedlabs/oracle	x/oracle/types/oracle_vote_callback.go	b1e5d65a74bfe088b802d77232adb024cda8052b64109b3f17b91a6da621b058
ORO	refractedlabs/oracle	x/oracle/types/oracle_pre_vote.pb.go	f2608c170e84e6557692d850f8e53cb4b6ec7c3d9e15aeef3c9475989c24fe98
PAL	refractedlabs/oracle	x/oracle/types/params.pb.go	85dec1ba41acdcde7ea2be6d53e7c3fbc8f9e1fc5e34a962a80f7ccbfbfe1f5a
QUS	refractedlabs/oracle	x/oracle/types/query.pb.go	0fc30619519546fcb99332fa4f5f75608a8c6350627d50a6f460b5f68bda37a1
QUW	refractedlabs/oracle	x/oracle/types/query.pb.gw.go	d1b03e4fe07d1a7bb5cf8065f0bbe20d50fe93bca8aab45a7db8c72a442429e3

ID	Repo	File	SHA256 Checksum
TXY	refractedlabs/oracle	x/oracle/types/tx.pb.go	7d1bfb4d0c729d0c8e7c5967c1dc9244463621682e2431bf748edb227476abe9
ORT	refractedlabs/oracle	x/oracle/types/oracle_vote.pb.go	f47180e704ef3beb035f07edaee225e5c27b0137f1b704582a074a7c31b6496e
MIO	refractedlabs/oracle	x/oracle/types/miss_counter.pb.go	277c960ccd2f6a650b2fae3f016e35ad04f2771ea1db6df409012cc93d6b217b
GOE	refractedlabs/oracle	go.mod	7e3f48e8350f81941225704f2ddba01030eda7d902fceca280870607bd42a715
ENC	refractedlabs/cosmos-utils	app/encoding.go	5e3679c70393a83f61a63ab50bce3b80b7f382e5a770690899f3e8eb063fac43
ANE	pryzm-finance/pryzm-core	ante/ante.go	91b10ca8346fd94a4f8142f2618269bfc807c8a4c9c7e2626a281fc53ae129ab
END	pryzm-finance/pryzm-core	app/params/encoding.go	e1b66751bc0c2fad0ab27432d0815223dc5dfc1207ad2e28259df6141f769f82
APA	pryzm-finance/pryzm-core	app/app.go	238b688d5ba06689af3b0178eb98323bd0afab109536e78a59efa325c29c03ce
AST	pryzm-finance/pryzm-core	app/asset_manager.go	b8943515b22df0284de2c6870182556507ca3927e13c550d303f8febdc75ae65
ENO	pryzm-finance/pryzm-core	app/encoding.go	2ef97bd4eaa6a7e68042f246252baec6a5ab1ab45864d4ee8c90e69bdeb0a8ae
EXO	pryzm-finance/pryzm-core	app/export.go	f0ca42dea27987807240aa2d1fbe94c34112e52a7b4e4de1241bef70b9f67e24
FEC	pryzm-finance/pryzm-core	app/fee_collector.go	366ebcdd8ad014c1e4f5623a2816fc1fb30151a287cbfcbad10db613652894b5
GE0	pryzm-finance/pryzm-core	app/genesis.go	fa720055e77331d79edf6ea877ea4cc3ab4cbee282681b10f77143e77f82b2c4
SIM	pryzm-finance/pryzm-core	app/simulation_test.go	fe1c99d512c57fcb55777b2139bd7b33f8a6696be895c96a94bd6cc31ae41a95
WAS	pryzm-finance/pryzm-core	app/wasm.go	5d9441817db99b89adc34e7347cff6f353d08b8f137394eea1e2bbce9d2e3413

ID	Repo	File	SHA256 Checksum
● ABS	pryzm-finance/pryzm-core	x/assets/keeper/abci.go	41b7b501a8873bd873179748e8976699fe0439efdde41ccd4be5cec6883efc7f
● EXN	pryzm-finance/pryzm-core	x/assets/keeper/exchange_rate.go	4ad13c420cab020bc782b0754f3bb58678a115b7988c00e8ee2a98d6ecd4ed44
● FET	pryzm-finance/pryzm-core	x/assets/keeper/fee_ratios.go	5a196f8d948eb0ddcdd617e4f066b9aa54757faf305d9a90e10cc1fe92491f7b
● GE5	pryzm-finance/pryzm-core	x/assets/keeper/genesis.go	4e302d93d90834670187cfb4bbb354f51b7bb3364f66d5a66e439b1b95cc5669
● GP5	pryzm-finance/pryzm-core	x/assets/keeper/grpc_query.go	1ac9fdce2c3fc78a457c0465988b8f759972107f28917036d6cd7c6d133b6cab
● GPN	pryzm-finance/pryzm-core	x/assets/keeper/grpc_query_exchange_rate.go	6a968c6dc0d6654f06df14c92c56833bb2962c45c2cb1a1f00535044296ba3c5
● GP9	pryzm-finance/pryzm-core	x/assets/keeper/grpc_query_maturity_level.go	49b0c2adb1b1daa5068483f290213954a80f84f96cc5be0cb1272d20d0081268
● GP6	pryzm-finance/pryzm-core	x/assets/keeper/grpc_query_params.go	1d8cc8c79747e2f233a31e230166da88e80da2ddc94ad50e7fb3e656d16a4913
● GP2	pryzm-finance/pryzm-core	x/assets/keeper/grpc_query_refractable_asset.go	16510e92f13c89af90b4921fd0c689a85fdcbaa5e26c06e16e0740e564107cd3
● ICT	pryzm-finance/pryzm-core	x/assets/keeper/icstaking_listener.go	57578af0bca466b0cf79fb48fdd54591054e107ee537c3f9cdffb4ba345b2ac9
● KEZ	pryzm-finance/pryzm-core	x/assets/keeper/keeper.go	cdb65110dd4efe00ae00ffb3572381121d21fde95d04c57908a394c694c996cc
● MTU	pryzm-finance/pryzm-core	x/assets/keeper/maturity_level.go	b8f064e295d525acf0230f713bcad4b5f9772ca8a4955a3bea1f9ca2f52d2a0f
● MR4	pryzm-finance/pryzm-core	x/assets/keeper/msg_server.go	d85043aa23e13109014111a4e40d92b439d947f287052a5f6ee7b161425bfd26
● MRI	pryzm-finance/pryzm-core	x/assets/keeper/msg_server_disable_asset.go	f6b9c0a635cb50d2e2f4fb8a3a1bfe238e828237c0d1333c181d007d48815835
● MR8	pryzm-finance/pryzm-core	x/assets/keeper/msg_server_register_asset.go	6366bda318e436de414806d40ccda95ab4c56fad1fd57ccb9d61f30e9d7d70fa

ID	Repo	File	SHA256 Checksum
MR0	pryzm-finance/pryzm-core	x/assets/keeper/msg_server_update_fee_ratios.go	fddccdf5701eedf14af1124d2cbae87e2b47e8ad5882def82e1680d944de6f6d
MR5	pryzm-finance/pryzm-core	x/assets/keeper/msg_server_update_maturity_params.go	8c59d0aec2c03288359b4718e15124d4731707fa17bcbeb1b010dc1f3cf2cef2
ORS	pryzm-finance/pryzm-core	x/assets/keeper/oracle_callback.go	d3a7eb59b89e7502ed9d2f7789a2268a692b55c3d1e10b9ab5150d1bef7a1b4f
PA8	pryzm-finance/pryzm-core	x/assets/keeper/params.go	63a961bf89368c77da10f3366e32f14382cc127e269e9735a0969eec74d4da21
REA	pryzm-finance/pryzm-core	x/assets/keeper/refractable_asset.go	1aa61dab494dbced6ee8e0695db66fe0f2eab9dfd8f9ce25216c4a073144b6f0
MOL	pryzm-finance/pryzm-core	x/assets/module.go	6cdf95593d6a397f77b817101fba92a1d8d086b21e48d249cbab2b5e9f68fe67
ABT	pryzm-finance/pryzm-core	x/icstaking/keeper/abci.go	555cccdbaf46f96006215f082cbf6fd1b0d54712a39af8acbf27e97416b6ff9a
BRK	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge.go	d45820548e93f86dc91787976ed87af29abc0533188698be8d2c4614dba00ae4
BRO	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_compound.go	5eb196f4237d047b545b0babbc029874b52973ae5b8d452017c88b56ccbda76b
BRL	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_delegate.go	a4cc9514a91423db2469a17a40088c1c564e94d16e8fa21a5bd615b66c3b5c2b
BRA	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_delegate_transfer.go	e6fa4bf7be129f0d06b1656f429cc94cc763a72b8822a2caff8c7620238574ea
BRP	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_ica.go	1a9b6d488b6752655a794a6c73c929ec197e5e51e06be36f14f80f4dfaaf27c7
BRN	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_redelegate.go	cc750b843f9b07d493afa750b23812c2b1d2f4459be3e1687552f6a110fabe79
BRH	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_set_withdraw_address.go	f3747294c13f7cbd3bf219bd82f12d41c715ddf74811627d4df10e778bba7dec
BRX	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_sweep.go	443d1f13dee50063de51252364fd49467e568f6b9a37079129c70e93896c655b

ID	Repo	File	SHA256 Checksum
● BRF	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_transfer.go	1033efc7855ea9dd7f122279f93de2f11c9e817e0bfcc4568353f26f58c0b0c3
● BR3	pryzm-finance/pryzm-core	x/icstaking/keeper/bridge_undelegate.go	7bcf1634ed19f516bd2376a40af1e1d9b6df71095c470e1fed8b71b745ce15e6
● DEE	pryzm-finance/pryzm-core	x/icstaking/keeper/delegation.go	6c5dc4412495c645d244a19331ec6b8d3d5cdc76c715ae85698ef0a120c03ffc
● EPC	pryzm-finance/pryzm-core	x/icstaking/keeper/epoch_counter.go	989aabb700d37934fdb65783e69632e43173a266b87c53578b00daf7b3c4448d
● GE9	pryzm-finance/pryzm-core	x/icstaking/keeper/genesis.go	32ed8ee98eecc396dbf79835bb960b35cee60e13200bd990d919502a70c5d0e0
● GE6	pryzm-finance/pryzm-core	x/icstaking/keeper/genesis_test.go	07dfbee408390c99567684b803464afbc32e6e862085051a4e68e91a2fce9e34
● GPZ	pryzm-finance/pryzm-core	x/icstaking/keeper/grpc_query.go	134ad99eaf8c0f59c387fb223b07eb2c08b4e45f2c45ff243412fd200bf8e9b2
● GCQ	pryzm-finance/pryzm-core	x/icstaking/keeper/grpc_query_host_chain.go	4afdaf5a48392d4880e295214bd301ee4253a87c47d2a41dbee89831b607a5d3
● GCU	pryzm-finance/pryzm-core	x/icstaking/keeper/grpc_query_params.go	7fd1a798fe065f7516af62f45935b6be943bf74dbff14d2d4cb3cda2259c93ea
● GCE	pryzm-finance/pryzm-core	x/icstaking/keeper/grpc_query_undelegation.go	23c87628f302afe8fe7b01047fb3c39ff964db27e5b113da3eb1dfbad10ff51c
● HOH	pryzm-finance/pryzm-core	x/icstaking/keeper/host_chain.go	ba1549a9442c706621eb44be3be043a82b3706490bc4a9d615e3b6ab8b372c2f
● KPI	pryzm-finance/pryzm-core	x/icstaking/keeper/keeper.go	2ab7b3a930be1b5f373b1167aea781ee7ad19c86f10b19b7fbf210158d97ad1b
● KPC	pryzm-finance/pryzm-core	x/icstaking/keeper/keeper_ica.go	a622a9ff4a952c722e785f129ece18fed0a83b4aeb71732089e4d711285011ff
● KPN	pryzm-finance/pryzm-core	x/icstaking/keeper/keeper_test.go	50a462b01bfcbca61b66660c6e73acf386d3b0c439a3b53744e7d1633d236fad
● MR9	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server.go	7728999f07aa251e4831b52325131b0be7992096114077a3eeb13bda2ffee82e

ID	Repo	File	SHA256 Checksum
● MR6	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_instant_unstake.go	9dfd0be4fdc673412351b5e5836931fa84639f66994c05b3298450b338e4c290
● MRL	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_rebalance_delegations.go	e717bd120a90a01ea3a0eaa5cf59bfa2bfa77acb8a25a5fc374cdfe8cf0a90a7
● MRH	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_redeem_interchain_account.go	dc3632aae3fd87217ffae5d3d0ed797b3309f5af3ab5805fa0ce3651e64c8ccd
● MR2	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_redeem_unstaked.go	41e61f74bd5de4136d4b4547726e7819e9c34bd920cc2f232dc6b9a9ba134de3
● MRZ	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_register_host_chain.go	68a12fab80627e7a07454e3797e3b743fb5421869a5d97c7675df7c302ef7902
● MVE	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_stake.go	86e021597d777806b197e5b2982b852e8363b43e7282398b45155c1c1fc01bf6
● MVR	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_unstake.go	1eb429e3f1509a974ccbf6796add23c72934012e4b492b10bb786a2a15eda597
● MVU	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_update_host_chain.go	0cc3951f9c72dd79962dae68c6ce575e5228a3b87da6749c550d1efad5054910
● MVP	pryzm-finance/pryzm-core	x/icstaking/keeper/msg_server_update_params.go	3793833565e4e3980d50ca155b9de884fdc0b48ccc86029c200771240fe5fac1
● ORN	pryzm-finance/pryzm-core	x/icstaking/keeper/oracle_callback.go	e59442d082d91fa99bb0888a21dea016664a717c52608fd6982333cd71b73b14
● PAN	pryzm-finance/pryzm-core	x/icstaking/keeper/params.go	12281fbba26984d87117943bbc98e0f4c0a9528bede2237d529a8e85389d4c40
● REK	pryzm-finance/pryzm-core	x/icstaking/keeper/reply.go	68c7c8711a46ce1fb9d941b4f4c5aef8ee2622dfaf5dcdf4b9d0f89a3fd079cd
● UNL	pryzm-finance/pryzm-core	x/icstaking/keeper/undelegation.go	774d6d1ebfbdb9b76517731a5b2669617303b17ee44e481802b99f13fee2150a
● ABN	pryzm-finance/pryzm-core	x/incentives/keeper/abci.go	5dff37b0fdf330b3b8e392b31d16dfae3c282581eef2045220553f9f57ca3fe6
● BOK	pryzm-finance/pryzm-core	x/incentives/keeper/bond.go	8a909b962af5bf897d2e800bf40d3254528f7aa8ca73f59d104898ae81ff9e1e

ID	Repo	File	SHA256 Checksum
● GE2	pryzm-finance/pryzm-core	x/incentives/keeper/genesis.go	feb278aecb619ec18b29e252c93a041dd019517fc0b4d0738e0a632c23221bdf
● GCR	pryzm-finance/pryzm-core	x/incentives/keeper/grpc_query.go	62126e436840433a7ba5a8adf3b1f2eb27e2d5cbc0962ccd633b3805fe096124
● GCY	pryzm-finance/pryzm-core	x/incentives/keeper/grpc_query_bond.go	29cabc72fc88a184e6efa0a29e77e8a9c5602812d9e12a8498c234e0eed6fbe4
● GCP	pryzm-finance/pryzm-core	x/incentives/keeper/grpc_query_params.go	76f3b949c3687ae93408c9bbd1677fb29866f83c57ad81d197ecd2f87234bffe
● GCO	pryzm-finance/pryzm-core	x/incentives/keeper/grpc_query_pool.go	056aaa1a8f3f6ef418d10b43a2e36d8c066f53d9aedec316b43bc3ccdd718b9d
● GCN	pryzm-finance/pryzm-core	x/incentives/keeper/grpc_query_unbonding.go	b0bf52b52fd46a4831b055a215ac623b313be8d1385bfb3388efd9ff6c6c1216
● INE	pryzm-finance/pryzm-core	x/incentives/keeper/incentivize_pool.go	058de9acc6f8db5ae4f7047166abfc6923ebf6e851a5c5bc19a2b156d55a0424
● KPV	pryzm-finance/pryzm-core	x/incentives/keeper/keeper.go	657ad75262bde9ac7f7d79951af42aec154adc6e33135fc44e60aaaac64452dc
● MIH	pryzm-finance/pryzm-core	x/incentives/keeper/mint_hooks.go	b346588b357c1d13dccbb7035cd0059a20073e4444c02a6c55391d0ad5413cc1
● MVK	pryzm-finance/pryzm-core	x/incentives/keeper/msg_server.go	f6f79b7b32d85cff070b38a1b1dc091d0c56bc010e87b24cf7096d97f1095957
● MVB	pryzm-finance/pryzm-core	x/incentives/keeper/msg_server_bond.go	c8e54fd2f7a4676155db040f458c0a296b077cda537f6ee3bd3c26009afd6665
● MVI	pryzm-finance/pryzm-core	x/incentives/keeper/msg_server_incentivize_pool.go	5652ed64d791407eedaa95dfa97fe9f86f132a33393c63b7592617effb6edff7
● MVO	pryzm-finance/pryzm-core	x/incentives/keeper/msg_server_pool.go	231266ffa6512461b5df0c60c9bd0f02ea965477cc10063b9590c75e5c566463
● MVL	pryzm-finance/pryzm-core	x/incentives/keeper/msg_server_pool_test.go	bdb6fc6e5faf0d48b9eb27bf929aec5b0008d02b6b63954db6a0b6ff35e2a1c2
● MVN	pryzm-finance/pryzm-core	x/incentives/keeper/msg_server_unbonding.go	cf01cce73105e0fa2a95e4946e8de93ba8a0f561604db0d55aae0ef6d0825e55

ID	Repo	File	SHA256 Checksum
● MVD	pryzm-finance/pryzm-core	x/incentives/keeper/msg_server_update_params.go	3d0873e83770b88efa58ec6b1d150921f54f1f1684673dd8e26666d38148336d
● PA0	pryzm-finance/pryzm-core	x/incentives/keeper/params.go	f5e2a6fe76356ef793fb7672442e5a4883616296744fa25e0e89d1f67997c894
● POP	pryzm-finance/pryzm-core	x/incentives/keeper/pool.go	e6e4a156c5711dfeb011afb612f2da4c9a64c08a9c527a15ec1b43ddb6b940c4
● UNI	pryzm-finance/pryzm-core	x/incentives/keeper/unbonding.go	1735a0dac6c2de2318f8affc520c893f0938a70afdc0326e3c10cf6e25d59955
● ACN	pryzm-finance/pryzm-core	x/refractor/keeper/action.go	edb8a553b008300641da1a26209f21a0af2446d460e047784e9d7fa80fa4d965
● GEZ	pryzm-finance/pryzm-core	x/refractor/keeper/genesis.go	60bd2f7f42aa65a18bca1d3450e262d840d3f7bcb349882f6bfc8267e0d7c8bd
● GCK	pryzm-finance/pryzm-core	x/refractor/keeper/grpc_query.go	c3c2e44c59401a3b2288b6b05d469fa27a02620f5914cf61ddd9ce520f054c7e
● KPO	pryzm-finance/pryzm-core	x/refractor/keeper/keeper.go	9e0472792c6a31b327304f761c08efc25daf7ca86c837f4b5898a00160b36281
● KPX	pryzm-finance/pryzm-core	x/refractor/keeper/keeper_action.go	38a15b25aa74b5080299188c59779e7ccb709bca0bba7798e076d511d62e6787
● KP3	pryzm-finance/pryzm-core	x/refractor/keeper/keeper_asset_state.go	aaabf2dbab976b17a2f433becfb77a278ceb6f9deaec59bb2f3a6db92b32f456
● KPB	pryzm-finance/pryzm-core	x/refractor/keeper/keeper_distribution.go	d9d6bacaccce2b33910e7f2dca452b8fe29923d61706e817ff8dba5f4d4f0b7b
● KP7	pryzm-finance/pryzm-core	x/refractor/keeper/keeper_redeem.go	79caead7dd2f812f753fbca1329380903e0145a446b80b648ede06e4f0be9369
● KP4	pryzm-finance/pryzm-core	x/refractor/keeper/keeper_refract.go	640c271f9eac804479f3a2c4c1983cb4c89bbbd12f60788d7fa4f0a0488bd645
● KP8	pryzm-finance/pryzm-core	x/refractor/keeper/keeper_test.go	0de7578571cb0182d3350c999e5aec241791a5c128c6f81074fe66a8b0b2edb5
● MVF	pryzm-finance/pryzm-core	x/refractor/keeper/msg_server.go	dfbe3785f430cfa0deb3c5188bcfc361be524aef990cb166786702bfa1a83f46

ID	Repo	File	SHA256 Checksum
MVM	pryzm-finance/pryzm-core	x/refractor/keeper/msg_server_redeem.go	6596dfd4d3c6f058e0447931d3d28ae43bf8ac423ef791a806e817c2bd4bf643
MVA	pryzm-finance/pryzm-core	x/refractor/keeper/msg_server_refract.go	44e3f02f4fdd08a04f005ecb40ab34d6aa5ad4d117efd916710a8c57b2b1e22c
ORF	pryzm-finance/pryzm-core	x/refractor/keeper/oracle_listener.go	6c497218b7d47927b19a613b1805ec4d0cb3ce6990d5b8b4bb3167bc784cc245
DAS	pryzm-finance/pryzm-core	x/mint/keeper/dapp_spend.go	148706c28cfe6eb96e2cd424650cf619535bddb10bdf0459bac6ace00ccb7c2c
DIT	pryzm-finance/pryzm-core	x/mint/keeper/distribution.go	e3efc0117b5e9e86420b2cb4cb5240dbe3fea8a2832144e3412082e0263a2017
GEH	pryzm-finance/pryzm-core	x/mint/keeper/genesis.go	4536686c6b2e5ed39a85cda12e12d456db9a7aabc353a1bae50f526918c64775
GCM	pryzm-finance/pryzm-core	x/mint/keeper/grpc_query.go	92194ead5251df946af7a1912cbc890c0ec669a60beec8a3521c9579dbfab9a9
GCI	pryzm-finance/pryzm-core	x/mint/keeper/grpc_query_minter.go	acf7c648d72b3483e14c3f510ce0a519c4132d28f7d4ac6315187f4c16f403a2
GCA	pryzm-finance/pryzm-core	x/mint/keeper/grpc_query_params.go	bae752253bacba415bea62ccc8edd20200034d66ea5a92dbd3a82659c88446da
HOE	pryzm-finance/pryzm-core	x/mint/keeper/hooks.go	2bda3e41b06d073d2fb5a15a8975b0752d3dba52bc9618fed29508ed91b66250
KP0	pryzm-finance/pryzm-core	x/mint/keeper/keeper.go	810f6dc822accf88b3b232055827ca4b048619d038937c59397dc183e84bf130
MIK	pryzm-finance/pryzm-core	x/mint/keeper/mint.go	06450b0da9887326d82991fe55ead73730b992b59102d77340619dfe49d89b1c
MIP	pryzm-finance/pryzm-core	x/mint/keeper/minter.go	5229976196ac5418d81400d09c612bc9039fde44467895defaaad06641648a7f
MVT	pryzm-finance/pryzm-core	x/mint/keeper/msg_server.go	765e928b476a3a39b932f7c9828c568d5926e8313a15de5360c9483380421af8
MVC	pryzm-finance/pryzm-core	x/mint/keeper/msg_server_dapp_account_spend.go	67148df8d52b73328523cd023eb721d4f6478e63798f6cce3b28fb353ac88bff

ID	Repo	File	SHA256 Checksum
MVS	pryzm-finance/pryzm-core	x/mint/keeper/msg_server_update_params.go	023f01b71ae533b8e1006e77f24c10556ddffeece7978d8db8da4995ce7fa7bc
PAF	pryzm-finance/pryzm-core	x/mint/keeper/params.go	08467295814efa03dc779e0eee97cd2056a6928853ee7b2ab7bb3de2b1d539db
ABO	pryzm-finance/pryzm-core	x/pgov/keeper/abci.go	6a95ae32965c6ef32a31e1d6c589e5eb59354b15b88d9fd94ec8c23d1ed8ee94
BR7	pryzm-finance/pryzm-core	x/pgov/keeper/bridge_vote.go	53a322e5025956edbc84083e4cfd359aa0ea84bb065e9c5daf97082128f41a17
GEU	pryzm-finance/pryzm-core	x/pgov/keeper/genesis.go	3d91149eda5704c44e9d73e58e30911048d57ac285796d292cb9618b21f2399c
GCG	pryzm-finance/pryzm-core	x/pgov/keeper/grpc_query.go	ac3981909d7050d995e8ff645b9c87bf85a9d7cd72f99d5482ed9474d8e83ab7
GCS	pryzm-finance/pryzm-core	x/pgov/keeper/grpc_query_params.go	5c20dd436fe612a851969c10a9bd54fb153d7bafe01d137c36c934ae3e198de9
GCL	pryzm-finance/pryzm-core	x/pgov/keeper/grpc_query_proposal.go	24d1bcce030435304f2aba61aeb623e54f6d4d47eb8f5953c3bb4e4f8150dbb3
GCT	pryzm-finance/pryzm-core	x/pgov/keeper/grpc_query_staked_p_asset.go	d89f066dbd757bfad04720b1f260845e439619eeb0617f2a6a01c916371bdffa
GCV	pryzm-finance/pryzm-core	x/pgov/keeper/grpc_query_vote.go	36a254dfde51a50a5f5ce2907f0335112258d832d2bc7075f7ca4db65b181ec4
KPG	pryzm-finance/pryzm-core	x/pgov/keeper/keeper.go	ec82244ce7c7e0c3b51e8634bbe347aed3ae78ad741a9e855f7d9b294edfbb59
KPL	pryzm-finance/pryzm-core	x/pgov/keeper/keeper_proposal.go	4563920b089ff95d3c57bc6f63b31c3b30ff43bd9c0ffbece140734b60cbfaff
KP5	pryzm-finance/pryzm-core	x/pgov/keeper/keeper_stake.go	d42be4a30c9e75a3301aa892807a69a1e63305ea26c6f9365d6c4febc14f4d5d
KP9	pryzm-finance/pryzm-core	x/pgov/keeper/keeper_test.go	81aed6193434ddf636d05d312f6cedd3a44a4176ac4d97dcd844357fd5174366
MVG	pryzm-finance/pryzm-core	x/pgov/keeper/msg_server.go	b4484533632cebd1b2787daa4116fa937be9e8c68975550a058f3e8443900034

ID	Repo	File	SHA256 Checksum
● MVY	pryzm-finance/pryzm-core	x/pgov/keeper/msg_server_retry_vote_transmit.go	4270d74969aae0c282337b2722177a9f0113313580e444e49f7b3df93a2f6e75
● MVV	pryzm-finance/pryzm-core	x/pgov/keeper/msg_server_stake_p_assets.go	afded72e6cc6664a04523c5e20c4d0e7d42e98227e273f1e28b28ce4bc1a0e4a
● MVX	pryzm-finance/pryzm-core	x/pgov/keeper/msg_server_submit_proposal.go	4471379ae8f807667d261bffc253c1846c6b1c82677104f77974230d2b0cbc0f
● MV3	pryzm-finance/pryzm-core	x/pgov/keeper/msg_server_submit_vote.go	649e90208182b0614eb996ea08a6615be129072f73e4911883d7de8a2307b69f
● MV7	pryzm-finance/pryzm-core	x/pgov/keeper/msg_server_unstake_p_assets.go	2659bf6ba5b921e8c329e1404b9264ca142b178a21b0c2b1bf91045b8b8907cf
● MV4	pryzm-finance/pryzm-core	x/pgov/keeper/msg_server_update_params.go	d063cc4d2b457d5b9f9e263b25da252c1086942478e1dab7267c193f8b7e8d53
● PA5	pryzm-finance/pryzm-core	x/pgov/keeper/params.go	2d1eed85b8bce15b9323e1760685e7aa937432ab1eb921ec8a485b0b13d9c0f9
● PA9	pryzm-finance/pryzm-core	x/pgov/keeper/params_test.go	5367f9ef4f0de649bdbb0064075063adfc7e9832073f81900a8e93733690263e
● PRP	pryzm-finance/pryzm-core	x/pgov/keeper/proposal.go	4dbe2ab547bba5cacfe6ebb1f9f77b62ae5de4ca12474ab56f0e11c925389f9c
● STK	pryzm-finance/pryzm-core	x/pgov/keeper/staked_p_asset.go	c1ef85df11dc4424543da2753c424176f614e3de517a437a9e58f7b43de90abc
● TAK	pryzm-finance/pryzm-core	x/pgov/keeper/tally.go	bcddef659d2050ae9a7ce990320196cc4ad231734b61a2f3a4ecde5c2a4e853c
● VOK	pryzm-finance/pryzm-core	x/pgov/keeper/vote.go	3a80eba8303448161ead7c68026b376e04efac800fd7c65cec0ed9aaad5b51d3
● ABC	pryzm-finance/pryzm-core	x/treasury/keeper/abci.go	8720fb7db5ed42dcfba2fdaca757e84dbd1f0bebc753b60431bde99074538420
● ACT	pryzm-finance/pryzm-core	x/treasury/keeper/action.go	431e1445eb723a332c595601cc6d891586f184013405be4f472469ab762a22af
● FLT	pryzm-finance/pryzm-core	x/treasury/keeper/flow_trade.go	11feee56bbcdf28c25628ee037715325380af949b8ede232594105b13fdf396a

ID	Repo	File	SHA256 Checksum
GEN	pryzm-finance/pryzm-core	x/treasury/keeper/genesis.go	0185deb89dca0e3e92195c0a0ef6cbcb0358f69ce0f0232db11a753b679b4b36
GRP	pryzm-finance/pryzm-core	x/treasury/keeper/grpc_query.go	7a22725a570b48f3de41fd605898d4c284f5788b54b7eb931426d880c3c403bb
GRC	pryzm-finance/pryzm-core	x/treasury/keeper/grpc_query_action.go	7400ea9bef5a341f85aaeec398721b697ce03f488d514c6fe711edb9ff22e611
GRQ	pryzm-finance/pryzm-core	x/treasury/keeper/grpc_query_flow_trade.go	12b7b4aa7970418ff99ebefbf3f1c314c471b35281a4969f976683b33e1e9e60
KEE	pryzm-finance/pryzm-core	x/treasury/keeper/keeper.go	625a2e82060094a3fce8d95ce5bfe0b58cb3fd26767e2f5b39b200f978d90a97
KEP	pryzm-finance/pryzm-core	x/treasury/keeper/keeper_fee_payment.go	f8c912d110b17432ee6db7cb09732fe868296221cfc55f4e707a8531cc6d0950
KER	pryzm-finance/pryzm-core	x/treasury/keeper/keeper_test.go	77acc4c5e5c7360ba0b4a477f1d673e737a0fa2c431222e82b351d5d9ab42657
MSG	pryzm-finance/pryzm-core	x/treasury/keeper/msg_server.go	0578ef9162cb28ba25c29c742ba2eeb01ed35af082a17c5cebbe501b37c5935a
MSS	pryzm-finance/pryzm-core	x/treasury/keeper/msg_server_set_action.go	c03b05171f0936263b8c37c434d4ad5321362c571a438d65216da13cdbbb2683
MSE	pryzm-finance/pryzm-core	x/treasury/keeper/msg_server_test.go	83deb016addea6bd984cd7c3aca5d46993164cd2622db9a999d51cf674be58d1
GEE	pryzm-finance/pryzm-core	x/ystaking/keeper/genesis.go	1aa1f4260d84afd4b624d94c4f2aafae73c41beaef0d8470b4e2861aff44d3d7
GRU	pryzm-finance/pryzm-core	x/ystaking/keeper/grpc_query.go	52ba420d5069ef33225532db16665e4659a8c675cb1d09ce95227c67869653ea
GRE	pryzm-finance/pryzm-core	x/ystaking/keeper/grpc_query_bonded_amount.go	8a8f47b57162a3b3c4f312dae91f510bc25cb442cee8a5ea298f549c2b30c8b0
GRR	pryzm-finance/pryzm-core	x/ystaking/keeper/grpc_query_reward.go	9dc33356d7a4901b46dcd7aeb05e963c6d19334fc3d13428eb739181704eacbe
KEY	pryzm-finance/pryzm-core	x/ystaking/keeper/keeper.go	4f122867e06d65e039f16808d480c4a227f7a166df5769c986b051645af8ec43

ID	Repo	File	SHA256 Checksum
KEB	pryzm-finance/pryzm-core	x/ystaking/keeper/keeper_bonding.go	d1bf2f9bf51a9c1b4d99dd4a7944901059b5826388831ec438b4a1f5decc85d4
KES	pryzm-finance/pryzm-core	x/ystaking/keeper/keeper_test.go	497776ca8225b914efb7ceee0bedf330d2c8462508e843fa8a2a73a734e2816a
MSR	pryzm-finance/pryzm-core	x/ystaking/keeper/msg_server.go	30f2b31b08e6d12c812e0fc39cca489982835c05173b607bc50849a79e0cc8f6
MSV	pryzm-finance/pryzm-core	x/ystaking/keeper/msg_server_bond.go	cd4ee7c35319eaf6f150d0bb3923c994675aa45b529112e6a2361654e573a90e
MSI	pryzm-finance/pryzm-core	x/ystaking/keeper/msg_server_claim_reward.go	4b32eec327d9ff7e80157b88520fe0929df829cfc04c821be3b4f2ef394878b1
MST	pryzm-finance/pryzm-core	x/ystaking/keeper/msg_server_exit_pool.go	fbefc74ec8be517c057d86aeb9f2876e17669abd286e2235a884f7d08cec4bd8
MSU	pryzm-finance/pryzm-core	x/ystaking/keeper/msg_server_unbond.go	0bb598dfde14655ac4a7a4bebaa0088c181b28ac7aea897f94816d708f5358a4
YIE	pryzm-finance/pryzm-core	x/ystaking/keeper/yield_listener.go	870e1dafce77eb86ea13801f213bf999c6a682f13ef455deee11911f625e8342
MAT	pryzm-finance/pryzm-core	x/ystaking/keeper/assetpool/maturity_yasset_pool.go	e6f381d4b2cd5c4f19e2106719edd65a79dbdbf655aab3abe5876f82751db9be
POO	pryzm-finance/pryzm-core	x/ystaking/keeper/assetpool/pool_store.go	03547886365d680ed0fd8288913ef643b5826d4c5d3731b21e15b727879ad1d8
YAE	pryzm-finance/pryzm-core	x/ystaking/keeper/assetpool/yasset_pool.go	f7b138f17345e8421a0dbc6df1e74790ea40b0c939ad971c902702bb5dd66c62
YAT	pryzm-finance/pryzm-core	x/ystaking/keeper/assetpool/yasset_pool_base.go	3f006cb635978d3b82bb141a0afb5e617d417e595a840a244e1edfd622034d7b
ABK	pryzm-finance/pryzm-core	x/amm/keeper/abci.go	30afb51c6dda5f315daf05796e95c1c6768e12eb960f9321cb38a9a6f7d64025
EXE	pryzm-finance/pryzm-core	x/amm/keeper/executable_order.go	821309f53c170a2209f3e4b43be368430088d47f9d8f6fb20379644a73bd668b
EXP	pryzm-finance/pryzm-core	x/amm/keeper/expiring_pool_token.go	ce607696d640f82b9cd49a5105f6689f1eb5a94bfa72cca6a1e95ee3eb61414a

ID	Repo	File	SHA256 Checksum
GES	pryzm-finance/pryzm-core	x/amm/keeper/genesis.go	eec84f62b86f3d8ec3ad75566a31e4db38628e2025979604e88ddb6d88b2b626
GRY	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query.go	10a5632aa5bf4b50fd13e991972802735886f878eb5a45a729f65e9216ee53f8
GRA	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_executable_order.go	b54db48d4ac608bc97d681b15c3359de19ab4c32d4abd8c5a1ab127287e7bd4d
GRI	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_expiring_pool_token.go	14a8dce9a3debb149664fe9ca0e0f897d75d92e30df6c933559fd05b678c8c24
GRN	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_introducing_pool_token.go	cb67942ade794a0500a0bba8fc31cfc8939e03b93038815b1e55f39320bdd724
GRO	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_lp_token.go	a409ac22ae0b4a8c2fcec2805e03e264300e16a88fd81d8d28165d240931df22
GRK	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_oracle_price_pair.go	63ad9113edb1b9c6f770696a49043a4d54a70170724f3c480ffd238b3dddcf1c
GRD	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_order.go	c4cce15c9e3783d63283692b0cf8e88f33a2135bf405d9d15e6bf3e202cc21e6
GRM	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_params.go	8679c5877bf1c5f3c836bc78628f57e0d8607f41ca74de443e66507741bfcc8a
GRG	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_pending_token_introduction.go	252585475626f8a095a11ee32e783cbb3e37cf66982a5caf30b153e7431a60ef
GR3	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_pool.go	1bccc79e7e8172ebdb66f658a463f4abc21a67f65870f52a193e51751d8292e7
GR7	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_pool_token.go	f21b3677b09a4b7e2ff4da3b8a3cb514a4b90e316d75896aafe92f5f2e6c3e44
GRS	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_schedule_order.go	879b0ad559c8a073d4c3e944e8135ce087c91f927004402ae67208a65c653401
GRB	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_batch_swap.go	64ec64850a2daa6eeb0aa0767162b717f42ff8c6e91782f91cf180c1a0824266
GR8	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_exit_all_tokens_exact_lpt.go	d88868e4732550c384cfca9a5069f83917ea430448708b667ec5dc21e533499b

ID	Repo	File	SHA256 Checksum
GR0	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_exit_exact_tokens.go	4de335362d6c38c73ede02264677f75880b866618ecfcb3ba5060f841672b7a1
GR5	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_exit_token_exact_lpt.go	df3704aa8ae07179dfd4d6a821207c700bd631a3a307b48ce31886594efad0ae
GRZ	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_initialize_pool.go	39c743b1e8d21cd950044d20f8ee213c4668d63341ddee650157af704a419ed4
GRJ	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_join_all_tokens_exact_lpt.go	81688a36562303ad7d52897b4331587931afa24717ac7ebb2c60d9bd57540a70
GR9	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_join_exact_tokens.go	866ebb197903ed94f346e6b5242a366ed0c7d7a51940c1127a847ff970db0c17
GR6	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_join_token_exact_lpt.go	382dfef395fc30f89cdb982b8dd4581222e0dd542387a22084e8a3f5ab52c23c
GRW	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_simulate_single_swap.go	c9a8f8012216188703533278 6ed920c51d22d532ae8648acdc6c06337787e11b
GR2	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_spot_price.go	b95270184b02cc646e90bffd15477fca37bc43907d3e6c0a09b5ef08a8978564
GRH	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_vault_pause_mode.go	c9cb079277b399af08921378aad9557f519310878cf0b18d7d98d14c0c39e650
GPC	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_weight_update_timing.go	12873a7232ecd4635ffc38c8874e27f39ac8c8bbfdb1fbdc3854f28f2bf5630b
GPQ	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_weighted_token.go	68a5078e37151a60d78904557785ba82c5d2376dd8db3db11a1a1b53501a4ae3
GPU	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_whitelisted_route.go	10503543cb6644b5380e59df9ac3beea4c49d21d5867a78b99a153bf0e40d4db
GPE	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_yamm_configuration.go	e2c5002dc8731a5519933ad4cdd085d79f86f744f4c28c4462ead18948a972fa
GPR	pryzm-finance/pryzm-core	x/amm/keeper/grpc_query_yamm_pool_id.go	334e6811d0f3eceea30d18ae8fc6aa3b00d05165e6fb9fc0ba77adf0d601c232
INT	pryzm-finance/pryzm-core	x/amm/keeper/introducing_pool_token.go	1ce9d3ae50dbb53517e0445b6fd4a4db38319b7c9ed221435604bfc0cb73e119

ID	Repo	File	SHA256 Checksum
KEA	pryzm-finance/pryzm-core	x/amm/keeper/keeper.go	f964769a74a6ad60f517e056dc12d896a5caa46d4866b47c128a09f566722dc1
KEM	pryzm-finance/pryzm-core	x/amm/keeper/keeper_test.go	28b5f02fb019ae01897f95995a77bc9e298226fe5e8f8bcac27d487469fa083f
MAU	pryzm-finance/pryzm-core	x/amm/keeper/maturity_level_listener.go	144e3dd09d92e7c59b6696593b8f7efae6d951b7339905010f6cc66949419648
MSP	pryzm-finance/pryzm-core	x/amm/keeper/msg_server.go	7157f14586a2f566da29a717acaab22f677129c194ce64950e8998ea93054c78
MSD	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_add_maturity_to_yamm.go	b00b2e8023bec8a0861cb8e90974fbf71cf633fdd4df0b1a938b4212fb28b765
MSB	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_batch_swap.go	13f82f9a71cf742c49185330d1f0a12906eff6c054f1762f4960320290cb8c0a
MSN	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_cancel_order.go	93221b54531b8700d20c62a974eb48d435015efadc39b02e167a23ebd2ba9a1c
MS3	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_cancel_pending_token_introduction.go	977cd4ab3d026d08b6038a43fdd6c960ca37ec24a900219d3541ddb7be8a6939
MSW	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_create_weighted_pool.go	2c74073def03d95d1419814f30158919fc97a5aef7ad198de2e6124e34e3d47b
MS7	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_exit_all_tokens_exact_lpt.go	424c0856e0cacb9224a70cdd58ec4afca26a41b313b35b8bd1d50e6d0003d8c5
MS4	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_exit_exact_tokens.go	09bb7f134b18718f59a604dcdaa2c6c7d1602e9f650ae3671198182d4d5df279
MS8	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_exit_token_exact_lpt.go	9fae67137b21653e5dd6e6a206210d124c807dd6c4b7f3e38a50bab9fb4ad1b1
MSZ	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_initialize_pool.go	31c7dff5e28153efdb60b3dd2c1fd0231583e3ab34bc87ad850c8de7297ac0ce
MSY	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_introduce_yamm_lp_to_weighted_pool.go	9ebca3ac12b6b767cdfe03eb79877f23813003fc6b2769b386b508301531e599

ID	Repo	File	SHA256 Checksum
MS0	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_join_all_tokens_exact_lpt.go	8f58168b1c59a9a27d456b3743d3fe1058f57831f7ba4ab64a94827ec7445a5d
MS5	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_join_all_tokens_exact_lpt_test.go	23b23428f749bb02f2ebbcbdb162cfbad32618a20e62cad9dab5cb96dd9d2040
MS9	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_join_exact_tokens.go	d8250e7573021b76bbc4dee58bc21ebe514533b5aa2b29a9d579e3d313f8179f
MGR	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_join_token_exact_lpt.go	51b7d88ece271888e6409422519d6ceb1b5b5774d98322f3b1ff5826f1611070
MGV	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_oracle_price_pair.go	533856b9732c6fbdf99594d63a4ca0c232a7a1b8ccb07ad53f39af33012eeffd
MGP	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_propose_match.go	6ce45203203cb503906b9a8b1893c74b20cc838f4e02b749aea0a57811a3d7e6
MGC	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_recovery_exit.go	f1d93e603372a9b3cfb1f18dde5d6303522c89894ac6c42f6a5ea9e3d5018aed
MGM	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_remove_token_from_weighted_pool.go	7b7f93c3163edcd00062ec219040ce5c3a03abec3c53cb78500dfd15152ef1a4
MGT	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_circuit_breakers.go	9eed7a289c10690618f3f60f3bf87abb1b5becb959aed650258f67550af36228
MGI	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_initialization_allow_list.go	1da170563b076b40c4d0a51a9cef3fd9ef890310c05c5172b28f0058422d4a73
MGJ	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_join_exit_protocol_fee.go	e3478372298df1aaab8fb015f8130786ba202bc38a6a02d1f3b51549f92c738e
MGA	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_pause_mode.go	5a2e957c85f4e7e3978a65ea92ee49b1e970d425f51c24b692642ecac94e63c6
MGO	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_recovery_mode.go	03b3326e3aa2942937ad21c6ecb7b78fc1a763d8c95dc80ea28cda4d2aa95a76
MGW	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_swap_protocol_fee.go	9b299f4f71bd4c1dd8bce726af9dff705b91af3f921164cf87f0228209f909ec
MGU	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_vault_pause_mode.go	fb7c074fe99142c7497b3a2e5bc79d913c76e74bbbad10e4dce998b1ed1baa89

ID	Repo	File	SHA256 Checksum
MGH	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_whitelisted_route_enabled.go	6885c7aec8e93281f5d8fa9c469f41b63c714074fad56965317ca1b65b5940f7
MGY	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_set_yamm_configuration.go	201054662c72440a1e2e4cbb8ad2f9a89613899f180fcc2aaff47bf6cbcb3d9b
MGN	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_single_swap.go	2e5fc719d950a01e747df389188287e0592024c73720b85cbba19a187511c41a
MGB	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_submit_order.go	38dcb38a8ede5ece940937546b65cdd45c6d417d4010488bc0b2bf86e21eb54d
MGD	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_update_params.go	d06f1c995f12f0c97f51cf4cae67794617f7ad70245ed3069bcf5b3b58c6beb9
MGF	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_update_swap_fee.go	ad92349bb1840078e1f9f81c4cf4f107cde21c8de7173b4f33862adc65947496
MGG	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_update_weights.go	30420b65696772f5b1044f861fa22bebcf1d7a04ab6f502c1e637edbee352926
MGL	pryzm-finance/pryzm-core	x/amm/keeper/msg_server_whitelist_route.go	0be9c11ed0db607242de5b5a6a274d87be470f280142cb6e2d562763348dfc67
ORA	pryzm-finance/pryzm-core	x/amm/keeper/oracle_callback.go	2bc2499f5f0c6541d5f1bcaba61730d9c6b5ffb5d2fd52248df7c224379664be
ORC	pryzm-finance/pryzm-core	x/amm/keeper/oracle_price_pair.go	03aeb3ae8ba5431d8181775ce037823dc2d6ebc3f87228b9f1088964d112d2e8
ORR	pryzm-finance/pryzm-core	x/amm/keeper/order_execution.go	39c1ad5273361a7279b0569af8e1d1b3a0cb9ac8dc1e615834e64f29318b8619
ORM	pryzm-finance/pryzm-core	x/amm/keeper/order_matching.go	a32f289570ad45d01a9a2ded4667b7aabee8821f0f7dfb19138bc25c21f3ca73
PAR	pryzm-finance/pryzm-core	x/amm/keeper/params.go	81bf2c5e4a2e1bd019296a56417d8b7e7406c30c3ef671d43aba20d42455e0ce
PAA	pryzm-finance/pryzm-core	x/amm/keeper/params_test.go	99c8dff6d19d243cb7d60d0b1fd617f7e1e1d875fb632b5bcb5e7dfc2b392293
PEN	pryzm-finance/pryzm-core	x/amm/keeper/pending_token_introduction.go	8c55a01770005229994fc7ebb6ebd4317a19af00e0b538a1ce599f3e12d10809

ID	Repo	File	SHA256 Checksum
POL	pryzm-finance/pryzm-core	x/amm/keeper/pool.go	20ea7f0c12a546c487be8cfa8b1c5b04d215ce0e4c3d2cbd2a26a4709c136e1f
POT	pryzm-finance/pryzm-core	x/amm/keeper/pool_test.go	3352511b711ced00c4a2382a25afd11848104637c178712fdc53fa995a45ef20
POK	pryzm-finance/pryzm-core	x/amm/keeper/pool_token.go	8eb0698bf7542eff02ab485cda0f7a2447a3c668c1e126380731f15cae381763
SCH	pryzm-finance/pryzm-core	x/amm/keeper/schedule_order.go	3252eb5492afe3da255debfbef02de4b0dba454e804b1182c6046e07ce7eb5aa
UTI	pryzm-finance/pryzm-core	x/amm/keeper/utils.go	c8b39ddc70395c3af6fddc1c5a9321dd8be0fa50e832dec80894a330cd055eb1
VAU	pryzm-finance/pryzm-core	x/amm/keeper/vault.go	ac4a09163e5efb10670912e97115f13f1cfdcd531b486deb9bb8aadd64ed25da
VAL	pryzm-finance/pryzm-core	x/amm/keeper/vault_batch_swap.go	b432220cf95d1e21e9692ab8872e01886bd535b53c9d3fd8c6d7d7b9837de046
VAT	pryzm-finance/pryzm-core	x/amm/keeper/vault_exit.go	4e243f36206e255625164dc53473b2a97ab6d411468b9ec55e05aa7f079992d0
VAJ	pryzm-finance/pryzm-core	x/amm/keeper/vault_join.go	76c40102cf1276be2b10edd1e01a42d6f6356ad0b587be4169711bd98eba7b84
VAS	pryzm-finance/pryzm-core	x/amm/keeper/vault_swap.go	fccf261eff41f13ed2b12c6e459e95d85fe188879f6625c6e004539bafe266ab
VAY	pryzm-finance/pryzm-core	x/amm/keeper/vault_y_asset_buy_swap.go	0662f584b8b9716c1293e47d87946566174bbb04214d4651fb3f75200a49e632
VAA	pryzm-finance/pryzm-core	x/amm/keeper/vault_y_asset_sell_swap.go	5cbd99a52a0aed8003fd07b44bbc5a5cb5428f3ad075eaf6fc679cc686e1119b
WEI	pryzm-finance/pryzm-core	x/amm/keeper/weight_update_timing.go	5a89e7a3b35e29d9d12a043f60ffdee6de28f0e43c6aa7946fcb18d91c904154
WEG	pryzm-finance/pryzm-core	x/amm/keeper/weighted_token.go	9efa3f1e3078ee446453fb16b9d2fc439cc77d52afd5617bbc26e95f6cb31ca5
WHI	pryzm-finance/pryzm-core	x/amm/keeper/whitelisted_route.go	378c102788bf55de863145e0808e8f86543d5e0294e2376bb13b35bf6cc4ab44

ID	Repo	File	SHA256 Checksum
● YAM	pryzm-finance/pryzm-core	x/amm/keeper/yamm_configuration.go	feee96169c2156672f76f80861148a428b262be79d4e236eee9d1ebbc3b65750
● YAP	pryzm-finance/pryzm-core	x/amm/keeper/yamm_pool.go	4add294d4923123dfedc70f018acffd57370874d2a67464c2e36e76a59ca4b7f
● BUY	pryzm-finance/pryzm-core	x/amm/keeper/pools/yassetmath/buy_y_asset.go	b87739cba53e131545d7b8e0daf6358bcfc278d6736c1eef9c9554fb1f126711
● SEL	pryzm-finance/pryzm-core	x/amm/keeper/pools/yassetmath/sell_y_asset.go	069157eaf1803fb112da3c778c557d25a671e1c53a5d324503c9ce6228f9a768
● BAS	pryzm-finance/pryzm-core	x/amm/keeper/pools/base_weighted_pool.go	547418f11bfcf1484526f2c1d4bcb76ad96a8b28ba34f29dcdc670140e0638a6
● BAE	pryzm-finance/pryzm-core	x/amm/keeper/pools/base_weighted_pool_controller.go	8483c0647adf1a90670f6e802c6e09e124bcb9c94361427df158a9543771596e
● POA	pryzm-finance/pryzm-core	x/amm/keeper/pools/pool_api.go	6705124e8a896dade379af1b7663d3ad8ebb765f088b07fc698b6ca3f4580ed9
● POE	pryzm-finance/pryzm-core	x/amm/keeper/pools/pools_test.go	90af7f38a8e5ef558d1bb0015c16006cdfef95f925333e70e535aa10bce7b3eb
● WEH	pryzm-finance/pryzm-core	x/amm/keeper/pools/weighted_pool.go	11c1cd620b0ff47593f3f4ab5d0b6196e9782b6bd82d660e9a8f7696cb3d659a
● YAO	pryzm-finance/pryzm-core	x/amm/keeper/pools/yamm_pool.go	40378838253b7856fd17df7aa9a9952de5cc7b061d8d400142bb8c7f423e84e5
● CIR	pryzm-finance/pryzm-core	x/amm/keeper/pools/circuitbreaker/circuitbreaker.go	89edb4404bda2bdf206ab839199ed545fe0baac219c81030a8efb920b00308cb
● WET	pryzm-finance/pryzm-core	x/amm/keeper/pools/weightedmath/weighted_math_exit.go	6d46a5437833abc1b7e14dfb069f2ce21c43e60dd360a4d626b42fccf97b80f9
● WEE	pryzm-finance/pryzm-core	x/amm/keeper/pools/weightedmath/weighted_math_join.go	dd0232dfa56680120d9b1246d8eefb028222513b17d388d9def2d74b38a8aeae
● WED	pryzm-finance/pryzm-core	x/amm/keeper/pools/weightedmath/weighted_math_swap.go	00e387cd3360c6c2296ed70e3e6a162c3eef87478d45ea04835bbc994d2f16e9

APPROACH & METHODS | PRYZM

This report has been prepared for PRYZM to discover issues and vulnerabilities in the source code of the PRYZM project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the project against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar projects produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Testing the project against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially functions that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

OVERVIEW | PRYZM

Icstaking

The **Icstaking** module allows users to stake the IBC tokens (from a host chain) from Pryzm to a list of validators on the host chain via IBC. These validators are registered on Pryzm such that each is assigned an expected weight with a total weight equal to 1, which can be thought of as a portfolio

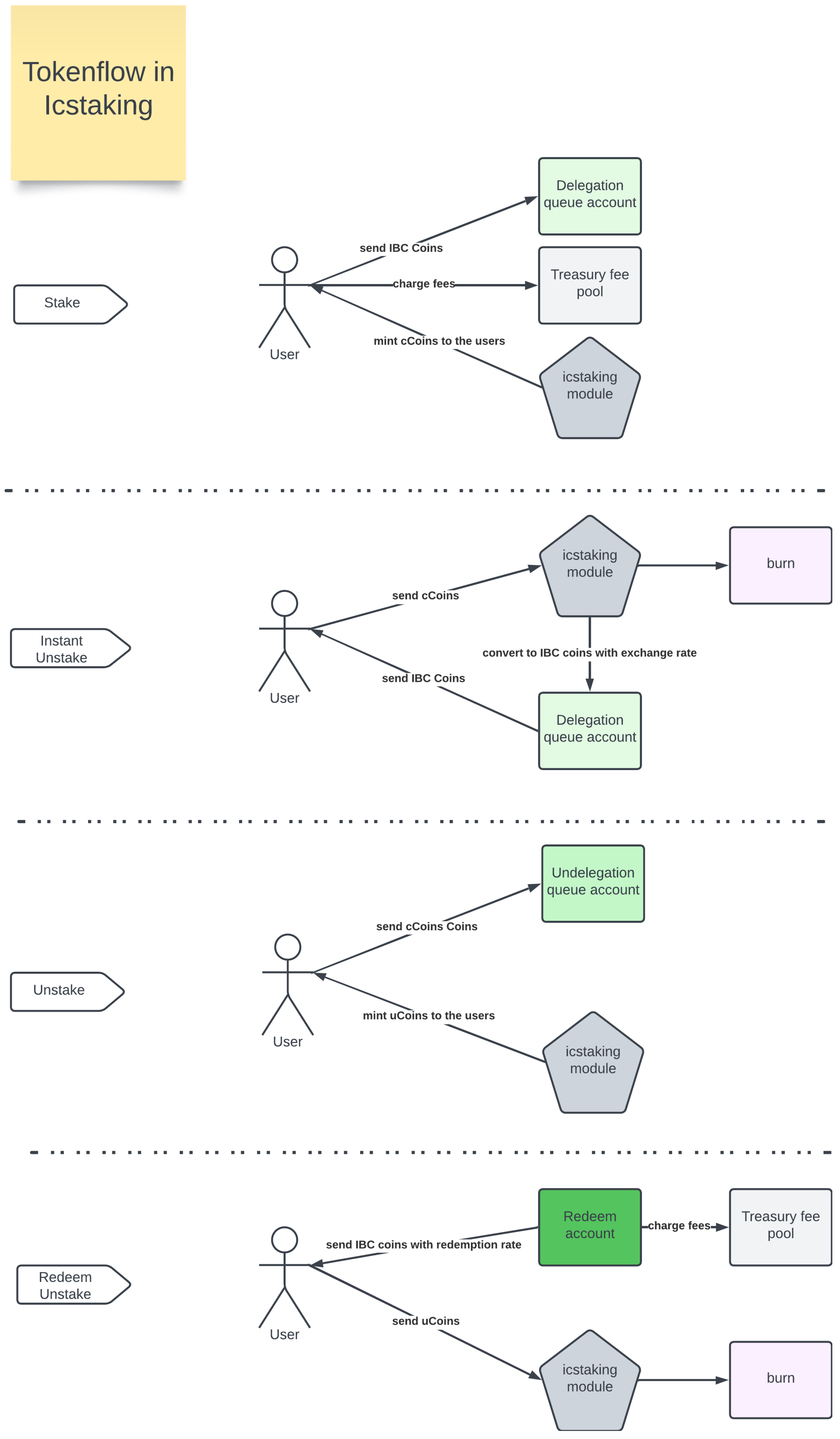
The users interact with the **Icstaking** module with the operations:

- Stake
- Instant unstake
- Unstake
- Redeem unstake
- Rebalance delegation

To facilitate the interchain staking, Pryzm creates the following accounts on the **Icstaking** accounts:

- **Icstaking module account** to mint/burn cCoins (wrapped token of IBC coins), mint/burn uCoins(accounting of undelegation with cCoins).
- **Delegation queue account** to accept/return the IBC coins from users.
- **Undelegation queue account** to accept the cCoins from users.
- **Redeem account** to receive the IBC coins from the host chain and send them to users.
- **Transfer account** to handle the IBC transfer between Pryzm and host chain.

The tokenflow in these operations on Pryzm is illustrated as follows:


Tokenflow in Icstaking
Stake
User
send IBC Coins
charge fees
mint cCoins to the users
Delegation queue account
Treasury fee pool
icstaking module
Instant Unstake
User
send cCoins
send IBC Coins
icstaking module
burn
convert to IBC coins with exchange rate
Delegation queue account
Unstake
User
send cCoins Coins
mint uCoins to the users
Undelegation queue account
icstaking module
Redeem Unstake
User
send IBC coins with redemption rate
send uCoins
Redeem account
charge fees
Treasury fee pool
icstaking module
burn

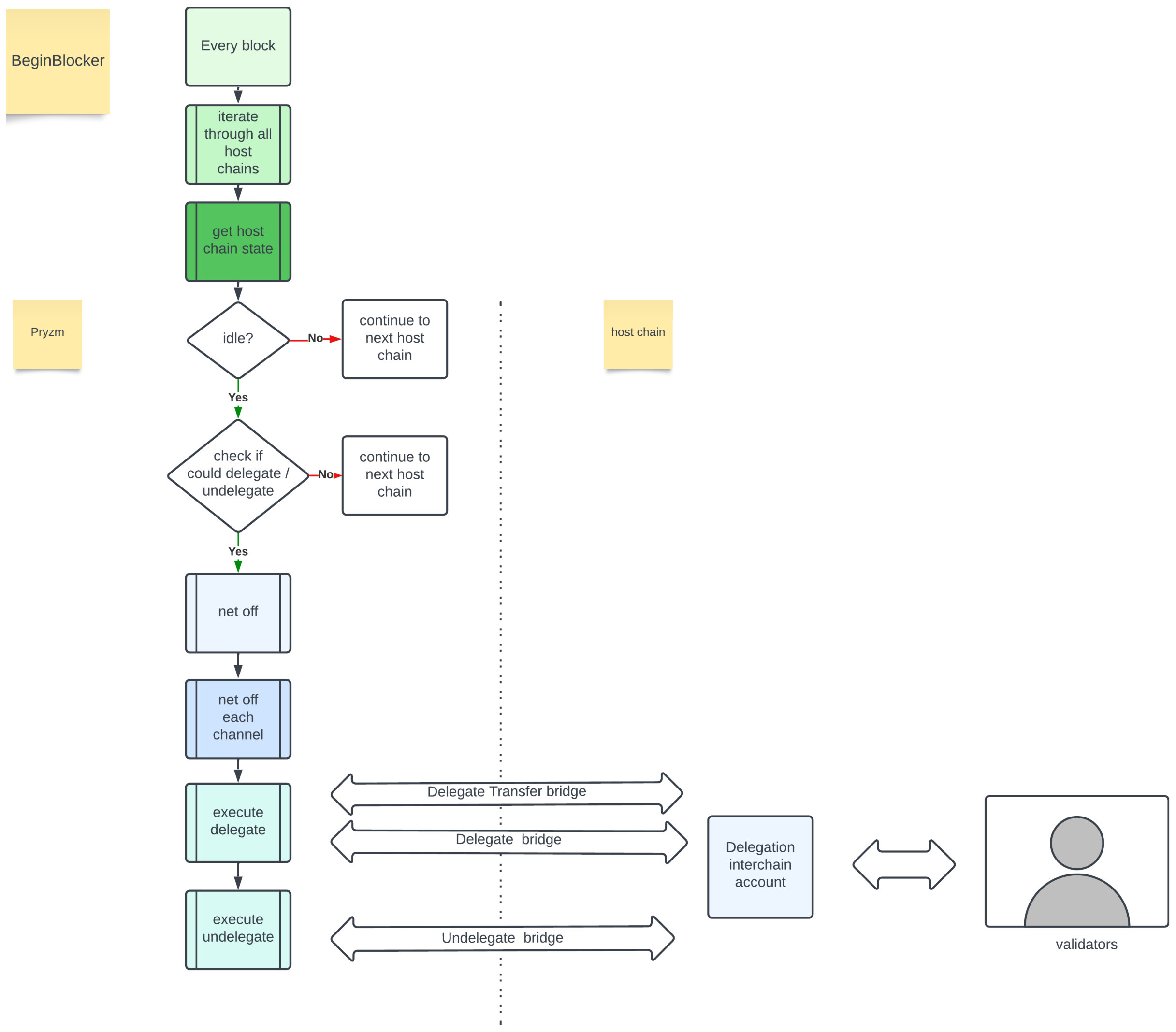
On the host chain, the following interchain accounts have been registered with Pryzm:

- **Delegation interchain account** to accept the host chain’s native coins (convert from the IBC coins) from the transfer account on Pryzm and delegate them to the validators on the host chain, undelegate the delegation, and receive the native coins from validators, and redelegate the delegation among the validators.
- **Reward interchain account** receives the staking reward from the validator, some of which will be sent to the delegation interchain for compounds, and the remaining part will be counted as the protocol fee and sent to the sweep interchain account.
- **Sweep interchain account** to receive the undelegated staking from the delegation interchain account and fee from the reward interchain account, then send them back to the transfer account. Then, the transfer account sends the IBC coins to the redeem account for the users' redeem.

BeginBlocker

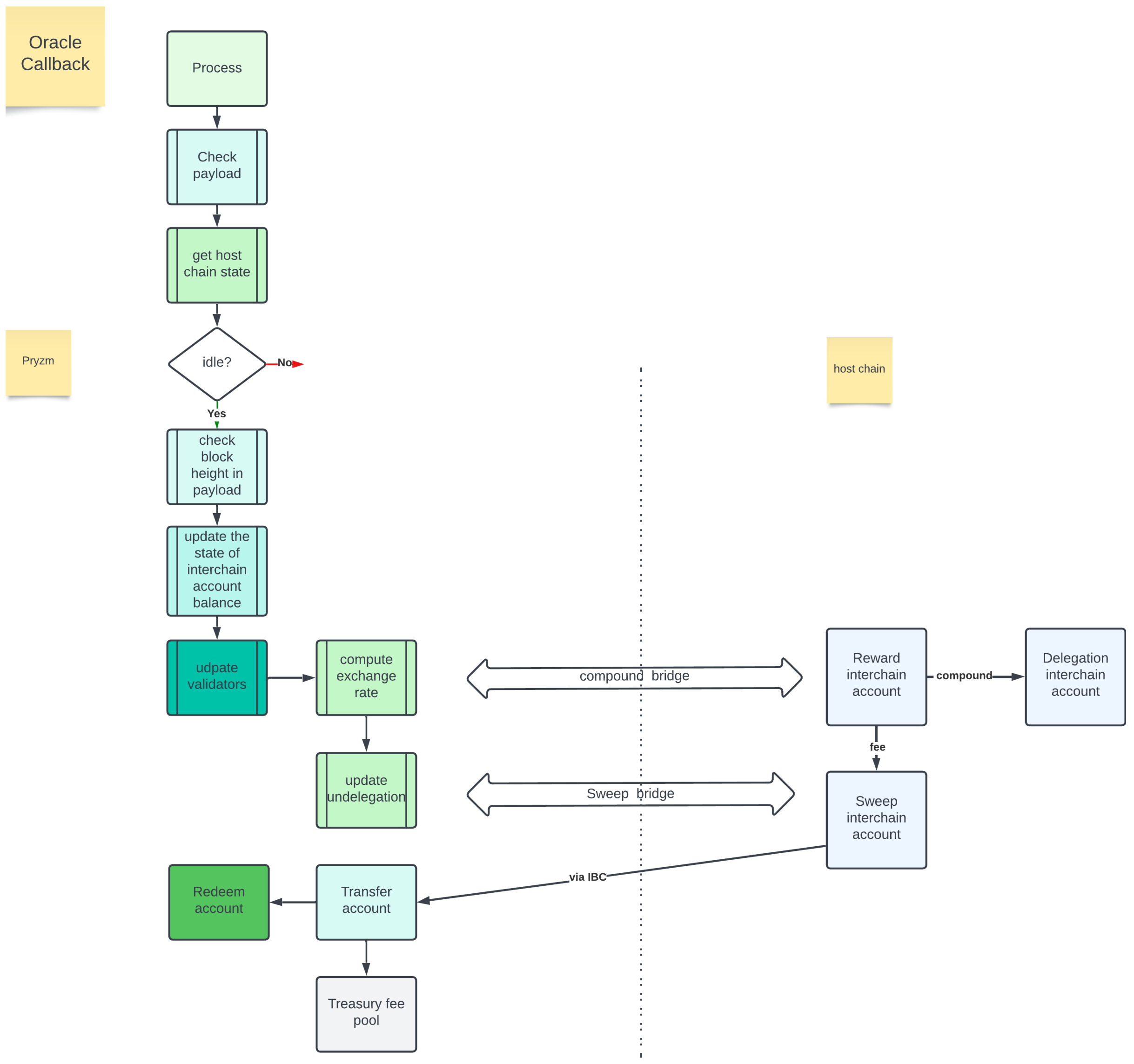
In the begin block ABCI call of **Icstaking** module, the delegate and undelegate will be executed for each host chain. It fetches each host chain’s state and checks the host chain’s state and if the delegation time and undelegation time have passed. If the next delegate or undelegate is ready, it attempts to net off the delegate amount with the undelegate amount for each channel on the host chain, which invokes the following 3 bridges to connect with the delegation interchain account on the host chain to complete the process:

- **Delegate transfer bridge**
- **Delegate bridge**
- **Undelegate bridge**



Oracle Callback

The state update of the host chain is reflected on Pryzm via the Oracle Callback.



FlowTrade

The **FlowTrade** is a module developed for swapping tokens during a defined period of time. Any user can create a flow of tokens by locking a deposit, distributed over a specific period of time, in exchange for another token. The tokens can be swapped continuously or discretely at a defined interval during the swap time. The price of the token is calculated based on the total amount of each token, and the time that has passed since the start of the flow. The token being sold, which is provided by the flow creator, is called `token-out` . The token provided by buyers is called `token-in` .

The users can interact with the **FlowTrade** module with the following operations:

- Create a Flow: Users can create a flow, by locking a deposit and providing some amount of tokens to sell.
- Join a Flow: Users can participate in a flow by providing any amount of `token-in` to the flow. Their share of each distribution interval depends on the time at which they provide the amount. When a user increases their position balance for a flow.
- Exit a Flow: Users can withdraw their remaining un-swapped `token-in` to exit a flow. When a user decreases their position balance for a flow.

AMM

The AMM is a module that supports various types of automated market makers(AMMs) and provides the ability to create, manage and interact with multiple types of pools. The amm module uses the vault to interact with all these pools. The vault holds all the tokens of all the liquidity pools and keeps track of the liquidity share of these pools by itself, users can do token swap operations via the vault, and the vault also provides batch swap operations to enable efficient pool interactions.

Another feature of the AMM module is the ability for users to leave long-term trade orders with price limits. These orders can be executed step-by-step to provide time-weighted pricing with lower price impacts. In addition to step-by-step order executions, the AMM module provides an order-matching system where proposers can propose to match a set of orders against the current price of the AMM.

Pools

Liquidity pools are the core building block in the AMM module, although the AMM module allows for various types of pools, all these pools share some common data and interfaces in the module. Pools can be established and overseen either by the government or, if permitted by the government, by an Externally Owned Account (EOA).

The Vault

There are multiple types of pools in the system and any number of pools of these types. However, the AMM module offers a unified method to interact with all these pools through a concept known as the Vault. The Vault serves as a unifying mechanism for this interaction. The vault holds all tokens of all the liquidity pools and keeps track of the liquidity share of these pools by itself.

Batch Swap

The AMM Vault offers a functionality known as batch swaps. A batch swap, as implied by its name, is a collection of swap operations that can be submitted to the AMM module to execute all at once. In the event that any of these steps encounters an issue, the entire procedure is rolled back, allowing the user to revert to their initial state.

There are two special types of pools implemented in the AMM module:

Weighted AMM Pool

A weighted pool is a special type of pool implemented in the AMM module, which uses a constant product equation and supports special operations such as updating weights of tokens, token swap, token introduction, and token removal. The implementation of the AMM introduces a concept called virtual balance for the token to control the price of the token.

WeightedPool provides the main functionalities of an Automated Market Maker (AMM):

1. Users can provide liquidity to the Pool through the "join" function and earn a portion of the transaction fees generated during the Pool's daily operations.

2. Users can swap one token for another within the Pool. The Pool supports the exchange of these two specific tokens.

The WeightedPool allows users to participate in liquidity provision and token swaps, enabling efficient and decentralized trading within the Pool.

Virtual Adjustment Balance

To facilitate seamless transitions during the addition of new tokens to the pool and the removal of expired tokens, the concept of virtual adjustment balances is introduced. These amounts are added to the virtual balances of the AMM pool and are functions dependent on time. These virtual adjustment balances will typically be zero and only deviate from zero when there is a need to add a new token to the pool or remove an existing one. In such cases, the corresponding virtual adjustment balance will gradually increase or decrease.

Liquidity Deposit

- Proportional all asset-deposit: All-asset deposits are executed proportionally, following the standard procedure. In this process, a user deposits an amount of each asset into the pool, and the deposited amounts must be proportionate to the actual balances in the pool. In return, the user receives a corresponding amount of LP tokens. The spot prices will not change after this operation.
- Single-asset deposits given LP amount: In this scenario, the user aims to make a deposit with a single asset token and desires to receive a specific quantity of LP tokens. To achieve this, the process involves conducting multiple trades of the token against each of the other tokens present in the pool, followed by a proportional all-asset deposit.
- Non-proportional multi-asset deposit: In this situation, the user intends to deposit variable amounts of one or more assets into the pool. To execute a proportional all-asset deposit, the user needs to engage in trades, exchanging appropriate amounts of some assets to acquire more of others. This ensures that the resulting asset amounts held by the user, post-trades, are proportionate to the actual balance of the pool. This operation involves a swapping step, and a certain swap fee will be charged.

Liquidity Withdrawal

- Proportional all-asset withdrawals: Proportional all-asset withdrawals follow the standard procedure, where users redeeming a specific amount of LP tokens receive proportional amounts of all the assets in the pool, corresponding to their share of the pool. As these withdrawals are proportional to all-asset withdrawals, the spot price remains unchanged after the liquidity is withdrawn for all assets.
- Single-asset withdrawals given LP amount: In this situation, the user intends to redeem a specific amount of LP tokens and, rather than receiving a proportional amount of all the assets in the pool, desires to receive only the token.
- Non-proportional multi-asset withdrawal: To perform this operation, the user can employ the deposit equations; they only need to input negative amounts for the specified values, and all other aspects of the process remain unchanged.

Token Introduction

If a weighted pool has been established and initialized, adding a new token with a zero balance directly is not feasible. To incorporate a new token into the pool, a virtual adjustment balance for the new token will be utilized, gradually decreasing over time. This approach initiates arbitrage opportunities, leading to a gradual increase in the actual balance of the added token.

Upon introducing the new token, it establishes its weight and adjust the weights of the other tokens in the Balancer-type pool accordingly. This adjustment is made in a manner that ensures the sum of the weights for all tokens in the pool remains equal to 1.

Token Removal

In the process of introducing new tokens into the pool, there may also be a need to remove existing tokens. To remove a specific token from the pool, virtual adjustment balances will be utilized once again. The impact of the virtual adjustment balance will cause the price of the token to gradually decrease, incentivizing arbitrageurs to buy the token from the pool. It is crucial to prevent traders from selling token to the pool, as doing so could potentially delay the removal of the token.

When the token removal function is invoked, the weights of the other tokens in the pool will be updated. This adjustment is made to maintain the sum of the weights for the remaining tokens in the pool, ensuring it remains equal to 1.

Yield AMM Pool

Another type of AMM implemented in the AMM module is a type of weighted pools specially designed for yield trading. With these pools the AMM module allows users to trade cASSET, pASSET, and yASSET of a specific asset per pool.

Yield AMM pools are specifically crafted to accommodate the unique characteristics of trading principal and yield tokens. Each refractable asset has its dedicated pool, comprising cAsset and active pAsset tokens as liquidity. Over time, as new maturity levels are introduced and old ones expire for the asset, the AMM module autonomously includes and removes the corresponding pAssets in the relevant pools.

Since the YAMM pools are based on the weighted pool designs, normal operations including swap, liquidity deposit, and liquidity withdrawal all can work the same as described in weighted pools.

YAMM mainly provides a way to exchange between YToken and CToken and has the following major exchange methods:

- doBuyYAssetGivenIn: The user is exchanging a specified amount of CToken for YToken. The Pool first borrows a portion of CToken, and then converts that portion of CToken and the CToken provided by the user into PToken and YToken using the refractor module. The Pool then converts the PToken back into CToken and uses that portion of CToken to repay the previously borrowed amount. Finally, the Pool transfers the YToken to the user.
- doBuyYAssetGivenOut: `doBuyYAssetGivenIn` and `doBuyYAssetGivenOut` are similar in that they both involve exchanging CToken for YToken. The difference is that `doBuyYAssetGivenIn` specifies the quantity of CToken input, while `doBuyYAssetGivenOut` specifies the quantity of YToken output.
- doSellYAssetGivenIn: The user is exchanging YToken for CToken. The Pool first borrows a portion of CToken and then uses that portion of CToken to exchange for PToken. The PToken and YToken are then exchanged back into CToken using the Refractor module. The Pool will use this CToken to repay the previously borrowed amount, and the remaining CToken will be transferred to the user.
- doSellYAssetGivenOut: `doSellYAssetGivenOut` and `doSellYAssetGivenIn` are similar in that they both involve exchanging YToken for CToken. The difference is that `doSellYAssetGivenOut` specifies the desired quantity of CToken to be exchanged, while `doSellYAssetGivenIn` specifies the input quantity of YToken.

Zero Impact Join

When a user intends to participate in a pool using only cASSETs, it is crucial to employ non-proportional join methods that involve underlying swaps. However, executing swaps during the join process may lead to undesirable high-price impacts for the user. To address this concern, a zero-impact join feature has been introduced. This feature enables users to join a YAMM pool exclusively with cASSETs, minimizing the price impact associated with the process.

The zero-impact join consists of two consecutive steps:

- Refracting a portion of the cASSET into the maturities present in the pool.
- Using the remaining cASSET and pASSETs to join the pool. As a result of these steps, the user receives LP tokens and yASSETs from the initial refracting process.

Order System

The order module provides a way to match trades. Users can submit their desired price as order data on the chain. At the end of a block, Pryzm matches all orders within the same trading pair (those wishing to exchange A for B and those wishing to exchange B for A). When the price and trade quantity are suitable, Pryzm processes these orders for a unified trade.

Incentives

The Incentive module can be divided into two parts: Pool management and Bond.

Pool Management

- CreatePool: Create a pool with the denomination and reward tokens array.
- UpdateRewardTokenWeight: Update the specified reward token's weight.
- AddRewardTokenToPool: Add a new reward token to a pool.
- IncentivizePool: Provide the rewards for the pool.

Bond

- bond: Similar to staking, users can send tokens to the module and get shares in some pool (depending on the denom of the token they bond).
- ClaimReward: Claim the rewards from the pool of the user bond tokens.
- Unbond: Like the withdraw function in the staking project. Users can redeem the tokens they bonded before. If the params.UnbondingPeriod is set to positive, the unbonded token will stay in the module for a while.
- ClaimUnbonding: Retrieve Unbonding tokens that have exceeded the time limit.
- CancelUnbonding: ReBond part of Unboding tokens.

Mint

Mint tokens and distribute them at the end of the epoch.

The epoch is specified by the `params.EpochIdentifier`. There are different types of epochs in the Epoch Module. The `AfterEpochEnd()` function will be called when every kind of epoch ends. It means that epochs can be measured in weeks, months, and years. If the code doesn't specify the epoch, the `AfterEpochEnd()` function may be called repeatedly because a moment may be the end of multiple epoch tickers.

The number of tokens that will be minted in this module is calculated by the following formula:

- mintedAmount = inflation * totalSupply
- inflation = inflation + inflationChangeRatePerEpoch * (1- bondedRatio/GoalBonded)

The totalSupply is the staking module bonded token totalSupply.

The minted token will be distributed to 5 addresses:

FeeCollector, Incentives, Oracle, developmentAddress, and DappAccount.

Oracle

Validators vote on the modules that need to call a callback. The system counts the votes at the end of each voting cycle and tallies the results. If the vote passes, it will call `oracle_callback.onMajorityVote()` on the corresponding module. If the validator has voted correctly then it will be rewarded. If the validator does not vote correctly, the system will accumulate a number of errors, and when the accumulated number exceeds a certain threshold, the validator will be slashed. If the validator does not vote correctly, the system will accumulate a number of incorrect votes. When the accumulated number of times exceeds a certain threshold, the validator will be slashed.

Assets

The **assets** module is designed to manage **refractable assets**, their **maturity levels**, and the **exchange rates** between each base asset and its refractable form.

The **refractable assets** are `CToken` (Liquid Staking Derivative) which can be refracted to `PToken(Principal Token)` and `YToken(Yield Token)` in the `Refractor` module. Besides the identifiers like unique asset id, token denom, and the host chain id of an external token, a **refractable asset** also manages its own `MaturityParams` and `FeeRatios`.

- The `MaturityParams` contains parameters to control the generation of `MaturityLevel`s for the asset. The first parameter `levels_per_year` decides how many `MaturityLevel`s a refractable asset has per year; the second parameter `years` decides how many years ahead of time the module will generate `MaturityLevel`s.
- The `FeeRatios` is used for managing the fee ratio of each operation, like protocol fee, fee for refracting operation, fee for staking YAssets, and so on.

The **MaturityLevel**s are auto-generated maturities for refractable assets. A `MaturityLevel` contains an `active` flag and the asset-id of the related refractable asset. Besides that, a `MaturityLevel` also manages the `symbol` , the `introduction_time,` and the `expiration_time` of a level.

For example, suppose the levels_per_year of a refractable asset is 2. In that case, the module can create a `MaturityLevel` whose `symbol` is "30Jun2023", the `expiration_time` is "20230701", and the `introduction_time` is the timestamp when the module creates this level automatically.

The **exchange rates** between each base asset and its refractable form. There are two ways to update the `ExchangeRate` : the first one is updating by the callback method of the `Oracle` module, and the second one is implementing the listener of the `Icstaking` module; the exchange rate will be updated as soon as the exchange rate updating occurs in the `Icstaking` module.

Only the `gov` module can interact with the `assets` module with the operations:

- Register new refractable assets.
- Disable a refractable asset. Please notice that once a refractable asset is disabled, there is no way to enable it.
- Update `MaturityParams` of a refractable asset.
- Update `FeeRatios` of a refractable asset.

Initialize Genesis State

In the genesis initialization, the default fee ratios will be set as the parameter of this module, if a refractable asset's fee ratios are nil, the protocol will use default fee ratios to collect fees.

The protocol also allows registering **refractable assets** and their **MaturityLevel**s and **Exchange Rates** in the genesis initialization.

BeginBlocker

There are two processes at the beginning of a block:

- The protocol will iterate all active `MaturityLevel` s and deactivate the expired levels(by setting the flag `active` to "FALSE").
- The protocol will iterate all enabled `RefractableAsset` s, create new `MaturityLevel` s for them and store the new `MaturityLevel` s. Please notice that the maturity generation for the current month will be skipped.

Listeners

There are two listeners declared in the `assets` module:

1. `MaturityLevelListener` : When `MaturityLevel` s are deactivated, or new `MaturityLevel` s are added, the listeners implemented in `amm` and `ystaking` modules will be called.
2. `ExchangeRateListener` : When `ExchangeRate` is updated in the `assets` module, the listener implemented in the `refractor` module will be called.

Pgov

ICStaked asset holders can participate in the governance of the asset's native chain. When users stake their assets in Pryzm's `Icstaking` module, their voting power is delegated to the delegation interchain account. The purpose of the `pgov` module is to enable users to participate in the governance of the asset's native chain by voting on proposals using their cAssets and pAssets on the Pryzm chain. To this end, proposals from supported asset chains are mirrored on Pryzm and voted on with a shorter voting period. The results are then submitted by an interchain message on the native chain.

The users can interact with the `pgov` module with the operations:

- Staking `pAsset` s. Holders of `cAsset` s can refract their assets to y & p assets and then stake `pAsset` s in `pgov` module to participate in native chain voting. The staked `pAsset` s are escrowed in `pgov` module account. The amount of staked assets is stored as a part of voting power for the holder. The assets can be redeemed anytime by the owner. But the amount of staked `pAsset` s at the time of tallying vote (After the end of the voting period) is calculated in the voting power.
- Mirroring Proposals from the host chain on the Pryzm network. The process of submitting the proposal on Pryzm is a customized type of **interchain query**. Anyone can submit a proposal from the target host chain to Pryzm, proof of proposal existence, and the height at which the proof is retrieved. Pryzm uses IBC light client state to verify the proof and stores the verified proposals with a proper voting end time.
- Voting proposals. Users can participate in voting as long as they hold `cAsset` s or stake `pAsset` s.
- Submitting voting results. After the end of the voting period (in EndBlocker), the protocol sends the voting results using icstaking bridges.

Refractor

The **refractor** module implements two functions：

1. Refracts a certain amount of `cAsset` s into `pAsset` s and `yAsset` s, and to redeem `cAsset` s.
2. Computes and distributes the yield to `cAsset` s held in the vault. The yield corresponds to the underlying assets of these `cAsset` s.

The users can interact with the `refractor` module with the operations, users may need to pay some fee for their operations:

- Specify a valid `Symbol` of maturity level, refract a certain amount of `cAsset` s into `pAsset` s and `yAsset` s. About the exchange rate between `cAsset` s and `pAsset` s(`yAsset` s), there are two cases:
    1. The specified `cAsset` is refracted for the first time in this module, the exchange rate will be the latest `ExchangeRate` which is updated by the listener of `assets` module.
    2. If there are refracted `cAsset` s in the module, the exchange rate will be calculated as $\frac{\text{total amount of pAsset}}{\text{total cAsset in module's vault}}$.
- Burn a certain amount of `pAsset` s to redeem `cAsset` s. According to the maturity level, there are 2 cases:

1. If the specified `pAsset` maturity level is not expired, the users must merge `pAsset` s and the same amount of `yAsset` s to redeem the `cAsset` s.
2. If the maturity level of the specified `pAsset` is expired, the users need to burn a certain amount of `pAsset` s and any amount of `yAsset` s to redeem the `cAsset` s.

Implemented `ExchangeRate` Listener

Once the `ExchangeRate` of a refractable asset is updated in the `asset` module and the new `ExchangeRate` value is higher than the old one, this listener's methods will be called to distribute yield corresponding to the underlying assets. The yield will be split into 3 parts:

1. Protocol fee.
2. Yield of staked `yAsset` s which have active maturity level.
3. The yield of unstaked `yAsset` s and the `yAsset` s which have expired maturity level, this part of yield belongs to the community.

Staking

The **ystaking** allows users to stake their yASSET and earn rewards through staked assets. The main functionalities are to enable users to participate in the staking process, withdraw their stake, and claim their earned rewards by `yAsset` s.

The users can interact with the `ystaking` module with the operations:

- Bond. Bonding is the process of staking an amount of a `yAsset` of a specific `maturity` by a specific user. Bond API is provided by `MaturityYAssetPool` , which can be obtained from the related `YAssetPool` by passing the `maturity symbol` .
- Unbond. Unbonding is the process of unstaking an amount of a `yAsset` of a specific `maturity` by a specific user. Unbond API is provided by `MaturityYAssetPool` , which can be obtained from the related `YAssetPool` by passing the `maturity symbol` . It is worth mentioning that unbonding is only available for active maturities and you must use `ExitPool` for expired maturities. When a user unbonds a value from a `MaturityYAssetPool` , an amount of unclaimed reward will be paid to the user.
- ClaimReward. Claiming reward is the process in which a user can claim all the accrued rewards for the bonded amount of a specific asset and a specific maturity. ClaimReward API is also provided by `MaturityYAssetPool` , which can be obtained from the related `YAssetPool` by passing the `maturity symbol` . It is worth mentioning that claiming reward is only available for **active** maturities and you have to use `ExitPool` for expired maturities.

Implemented Listeners

1. `MaturityLevelListener`

   For the `YStaking` module to deactivate maturity yAsset pools as the maturities are expired, this module implements the `MaturityLevelListener` of the `Assets` Module.

2. `YieldListener`

   For the `YStaking` module to keep track of yield and compute rewards, as soon as any yield is accrued, this module implements the `YieldListener` of the `Refractor` module.

Treasury

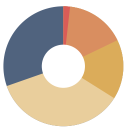
The **treasury** module serves the purpose of fee collection, and it also introduces a scheduling system that facilitates the allocation of the accumulated fees to specific tasks or purposes as required.

Within the Pryzm framework, every individual module possesses the capability to levy a protocol fee on its respective operations. These accrued fee amounts are consolidated within the treasury module's account.

The **treasury** module allows the governance to schedule the execution of certain actions on the collected fees. These actions can be one of the following:

- Hold: Do nothing and hold the collected fees.
- Burn: Burn Pryzm tokens, effectively removing them from circulation, and providing deflation on the native token.
- Distribute To Stakers: Send the Pryzm tokens to the feeCollector account, which is then used to reward delegators and validators.

FINDINGS | PRYZM



56	1	9	9	20	17
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for PRYZM. Through this audit, we have uncovered 56 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
YAM-01	YAMM Design Flaw	Logical Issue	Critical	● Resolved
BRI-02	Failure Of `HandleIBCTransferRecv()` Due To Incorrect IBC Denom	Logical Issue	Major	● Resolved
GEN-01	Unexported Expiring Token List May Lead To Users' Asset Loss	Coding Issue	Major	● Resolved
KEA-02	Potential Failure To Zero Impact Join The Yamm Pool Due To The Expiring Or Expired PAsset	Logical Issue	Major	● Resolved
KEE-04	PendingCAmount, PendingAmount, And ReceivedAmount Are Not Updated After Handling Undelegation Reception	Logical Issue	Major	● Resolved
KEE-05	Potential Panics When Fetching Nil Validator From `weightDiff`	Volatile Code	Major	● Resolved
KEE-06	Incorrect Distribution Of Delegation And Undelegation Amount As The Last Validator Gets Entire `remainingUndelegation` And `remainingDelegation`	Incorrect Calculation	Major	● Resolved
KEP-02	Potentially Unable To Exit The Flow Successfully Due To The Flow Being Stopped By The Flow Creator	Logical Issue	Major	● Resolved

ID	Title	Category	Severity	Status
KER-02	**Centralization Related Risks**	**Centralization**	**Major**	● **Acknowledged**
X0C-03	Potential Consensus Failure By Non-Determinism Of Map Iteration	Volatile Code, Denial of Service	Major	● Resolved
APP-01	Potential DoS Attack As Custom Module Accounts Are Not Initialized	Denial of Service	Medium	● Resolved
BRD-03	Incorrect Calculation Logic On `totalDelegation`	Logical Issue	Medium	● Resolved
BRG-01	Incorrect Update Of `hostChainState.AmountToBeCompounded`	Incorrect Calculation	Medium	● Resolved
CLC-01	Misconfigured Transaction Commands Are Blocked In Icstaking Module	Volatile Code	Medium	● Resolved
CLI-01	Misconfiguration Of Expected Arguments Blocks The Commands `CmdIntroduceYammLpToWeightedPool()` And `CmdSetJoinExitProtocolFee()`	Volatile Code	Medium	● Resolved
CLI-02	Missing FeeRatio Flag In The Commands `CmdSetJoinExitProtocolFee()` And `CmdSetSwapProtocolFee()`	Volatile Code	Medium	● Resolved
EXP-01	Failure Of Exporting Genesis File Caused By Fetching Validator Address Incorrectly	Volatile Code	Medium	● Resolved
MSG-01	Fee Is Collected From User's Address Instead Of Redeem Account	Logical Issue	Medium	● Resolved
ORA-02	Variable `hostChainState.AmountToBeCompounded` Used To Compute The Exchange Rate Includes Protocol Fee	Logical Issue	Medium	● Resolved

ID	Title	Category	Severity	Status
ABC-01	Heavy Computation In Icstaking's BeginBlocker Could Slow Down Block Production	Volatile Code, Denial of Service	Minor	● Resolved
ASS-01	Validation Of Genesis State In `assets` Module	Volatile Code	Minor	● Resolved
BRE-01	Discussion On The Delegation Rebalance Logic	Logical Issue	Minor	● Resolved
BRG-02	Non-Guaranteed Host Chain State	Logical Issue	Minor	● Resolved
CRE-01	Possible Overwrite Of Denom Metadata In Genesis	Volatile Code	Minor	● Resolved
CRE-02	Missing Display Denom Will Fail Denom Metadata Validation	Volatile Code	Minor	● Resolved
FLO-03	The Claimable Purchased Token Amount Does Not Consider `PendingPurchase`	Logical Issue	Minor	● Acknowledged
HOO-01	Mint PRYZM Each Epoch	Logical Issue	Minor	● Acknowledged
HOS-02	Return Value Of `GetChannel()` Is Not Handled	Volatile Code	Minor	● Resolved
KEE-03	Lack Of Validation For `transferChannel`	Logical Issue	Minor	● Resolved
KEE-07	Potential Division By Zero	Volatile Code	Minor	● Resolved
KEK-01	Incorrect Account Number Of `tokenfactory` Module Account	Inconsistency	Minor	● Resolved
KER-03	Lack Of State Validation For `WhitelistedRoute`	Logical Issue	Minor	● Resolved
POS-02	Potential Unable To Acquire `token-in` Tokens That Have Not Been Exchanged	Logical Issue	Minor	● Resolved

ID	Title	Category	Severity	Status
QUE-01	Incomplete Inputs Of Undelegation Query	Volatile Code	Minor	● Resolved
REF-01	Lack Of Validation Of The `RefractableAsset.FeeRatios` Field	Volatile Code	Minor	● Resolved
TYP-02	Missing Stateless Check Of `TransferChannel` In Messages	Volatile Code	Minor	● Resolved
VAU-01	Lack Of Minimum Liquidity Restriction In Pool Initialization	Logical Issue	Minor	● Acknowledged
WEI-02	Lack Of Check For Weight Update Period	Logical Issue	Minor	● Resolved
X0C-02	Potential Key Collision Because Denom Could Contain "/"	Volatile Code	Minor	● Resolved
ASS-02	Unnecessary Arg In The `QueryGetMaturityLevelRequest`	Coding Style	Informational	● Resolved
BAS-01	No Validation Of The Expiring Or Expired PAsset In Function `JoinAllTokensGivenExactLptOut`	Logical Issue	Informational	● Resolved
FLO-01	The Purpose Of The Deposit `creationDeposit`	Logical Issue	Informational	● Resolved
GEE-01	Missing Validation Of `ChannelUndelegationList` In Icstaking Module's Genesis State	Volatile Code	Informational	● Resolved
GLOBAL-02	Cosmos Messages Need To Extend `cosmos.msg.v1.signer` Option	Volatile Code	Informational	● Resolved
GO3-01	Insecure Cosmos SDK Version	Logical Issue	Informational	● Resolved
ICS-01	Typo In Message And Function Name `RedeemInterchainAccount`	Coding Style	Informational	● Resolved
KED-01	Discussion On `ExchangeRate` Updating And `YAsset` Yield Distribution	Volatile Code	Informational	● Resolved

ID	Title	Category	Severity	Status
KEE-08	Inconsistent Function Name `NewRedelegateMessageBridge()`	Coding Style	Informational	● Resolved
KEP-01	Discussion On Prices Of `token-in` And `token-out`	Logical Issue	Informational	● Resolved
MES-01	Missing Validation Of `epoch` In Message `MsgRedeemUnstaked`	Volatile Code	Informational	● Resolved
MIN-01	Discussion On The Calculation Of The Minted Token	Incorrect Calculation	Informational	● Resolved
MSG-02	Equality Could Possibly Not Be Satisfied Due To Rounding Issue	Volatile Code	Informational	● Resolved
ORA-01	Possible Increase Of Exchange Rate	Logical Issue	Informational	● Acknowledged
PAR-01	Typo In Error Messages	Coding Style	Informational	● Resolved
PRY-01	Gas Is Not Consumed If An Error Occurs Beforehand	Logical Issue	Informational	● Acknowledged
TOK-01	Incorrect Error Message In The Validation Of `CircuitBreakerSettings`	Coding Style	Informational	● Resolved

YAM-01 | YAMM DESIGN FLAW

Category	Severity	Location	Status
Logical Issue	● Critical	yamm_pool.go (374cad8)	● Resolved

Description

Files:

- `x/amm/keeper/pools/yamm_pool.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The design of YAMM uses the same mechanism as weighted pools but uses token weights depending on token balances. This change is conflict with the original weighted pool design and may lead to users' asset loss.

The weighted pools use a constant product equation:

$$\Pi_i B_i^{w_i} = C,$$

where B_i and w_i are the balance and weight of the ith token in the pool. The weights w_i depend on the time when there are weight updates or token introduction/removal. Otherwise, the weights w_i are constants.

Meanwhile, the YAMM pools use the same constant product equation with a different token weight definition:

$$w_1 = \rho(t),$$

$$w_i = \left(\frac{B_i}{B_1}\right)^{\alpha_i(t)} \text{ for } i > 1$$

Since the weight of the ith token depends on the token balance, users will have different strategies to add/remove liquidity or swap tokens, and different strategies will have different outcomes.

For example, if a user wants to swap x `tokenA` $(i = 1)$ for `tokenB` $(i = 2)$ in a YAMM pool, he/she might consider two strategies:

- **Strategy 1**: Swap x `tokenA` directly for `tokenB` . In this case, the swap amount calculation uses $(B_2/B_1)^{\alpha_2(t)}$ as `tokenB` 's weight.
- **Strategy 2**: Swap part $(\tilde{x})$ of `tokenA` for $\tilde{y}$ `tokenB` and then swap the rest $(x - \tilde{x})$ `tokenA` for `tokenB` . This is a two-step swap:

- In the first swap, the swap amount calculation still uses $(B_2/B_1)^{\alpha_2(t)}$ as `tokenB` 's weight.
- In the second swap, `tokenB` 's weight is changed to $((B_2 - \tilde{y})/(B_1 + \tilde{x}))^{\alpha_2(t)}$, which is different from the weight in the direct swap. Since the weight is smaller, the user can receive more `tokenB` than that in Strategy 1.

Therefore, users can choose their strategies to get different swap results or even **drain funds** from the pool.

Scenario

In the following scenario, a user swaps a token for another and then swaps back to drain funds from the YAMM pool.

1. Create and initialize a YAMM pool using the parameters in `x/amm/keeper/yamm_pool_test.go` .
2. Swap 2,000,000 `cAsset00` for 2,661,848 `p:Asset00:30Jun2024` .
3. Swap half (1,330,924) of received `p:Asset00:30Jun2024` for 1,063,442 `cAsset00` .
4. Swap another half (1,330,924) of received `p:Asset00:30Jun2024` for 939,855 `cAsset00` .
5. The total amount of received `cAsset00` is 2,003,297, which is more than the initial `cAsset00` amount 2,000,000.

Proof of Concept

The following unit test implements the aforementioned scenario. It works in `x/amm/keeper/yamm_pool_test.go` :

```go
func (s *keeperTestSuite) TestYammPoolSwapABA() {
    s.setZeroProtocolFeeParams()
    // maturities:  [cAsset, "2024/12/27", "2024/6/30", "2023/12/31"]

// weights:  ['1.500000000000000000', '1.001950944029203516',
'1.000000000000000000', '178.802575041768663606']

// normWeights ['0.008227991005099100', '0.005496028903348559',
'0.005485327336732734', '0.980790652754819688']

    pool, _, tokens, _ := s.createAndInitializeYamm(0)

    tokenA := tokens[0]
    tokenB := tokens[2]

    // Swap 2_000_000 tokenA for 2_661_848 tokenB.
    tokenASpent := sdk.NewInt(2000000)
    swapResponse, err := s.msgServer.SingleSwap(s.ctx, &types.MsgSingleSwap{
        Swap: types.Swap{
            PoolId:   pool.Id,
            Amount:   tokenASpent,
            SwapType: types.SWAP_GIVEN_IN,
            TokenIn:  tokenA.Denom,
            TokenOut: tokenB.Denom,
        },
        Creator: s.authority,
    })
    s.Require().NoError(err)
    s.Require().Equal(sdk.NewCoin(tokenB.Denom, sdk.NewInt(2661848)),
 swapResponse.AmountOut)
    tokenBReceived := swapResponse.AmountOut.Amount

    // Swap 1_330_924 tokenB (half of received tokenB) for 1_063_442 tokenA.
    tokenAReceived := sdk.ZeroInt()
    swapResponse, err = s.msgServer.SingleSwap(s.ctx, &types.MsgSingleSwap{
        Swap: types.Swap{
            PoolId:   pool.Id,
            Amount:   tokenBReceived.QuoRaw(2),
            SwapType: types.SWAP_GIVEN_IN,
            TokenIn:  tokenB.Denom,
            TokenOut: tokenA.Denom,
        },
        Creator: s.authority,
    })
    s.Require().NoError(err)
    s.Require().Equal(sdk.NewCoin(tokenA.Denom, sdk.NewInt(1063442)),
 swapResponse.AmountOut)
    tokenAReceived = tokenAReceived.Add(swapResponse.AmountOut.Amount)


// Swap 1_330_924 tokenB (another half of received tokenB) for 939_855 tokenA.
```

```
    swapResponse, err = s.msgServer.SingleSwap(s.ctx, &types.MsgSingleSwap{
        Swap: types.Swap{
            PoolId:    pool.Id,
            Amount:    tokenBReceived.QuoRaw(2),
            SwapType: types.SWAP_GIVEN_IN,
            TokenIn:  tokenB.Denom,
            TokenOut: tokenA.Denom,
        },
        Creator: s.authority,
    })
    s.Require().NoError(err)
    s.Require().Equal(sdk.NewCoin(tokenA.Denom, sdk.NewInt(939855)), swapResponse.AmountOut)
    tokenAReceived = tokenAReceived.Add(swapResponse.AmountOut.Amount)


// After swapping tokenA for tokenB, and tokenB back for tokenA, we will receive more tokenA than spent.

    s.Require().Greater(tokenAReceived.Int64(), tokenASpent.Int64())
}
```

Result:

```
Running tool: /usr/local/go/bin/go test -timeout 30s -testify.m
^(TestYammPoolSwapABA)$ github.com/pryzm-finance/pryzm-core/x/amm/keeper

WARNING: proto: file name query.proto does not start with expected testdata/; please
make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name testdata.proto does not start with expected testdata/;
please make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name tx.proto does not start with expected testdata/; please
make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name unknonwnproto.proto does not start with expected
testdata/; please make sure your folder structure matches the proto files fully-
qualified names
PASS
ok      github.com/pryzm-finance/pryzm-core/x/amm/keeper    1.231s
```

Recommendation

Recommend revisiting the design of the YAMM pool to avoid aforementioned situations.

Alleviation

[Pryzm Team - 09/15/2023]:

The team heeded the advice and resolved this issue in the commit `9de92bdfe6faf613480ccbba409ec89d87860cd9` .

[CertiK - 12/06/2023]:

In the new design, the token weight $w_j = k_j(t)$ for $j \in 1, 2, ..., n$ (i.e., pAssets), with

$$k_j(t) = \begin{cases} \frac{1}{1-\alpha_j(t)}, & \text{if } \alpha_j(t) \leq 0.98 \\ 50, & \text{if } \alpha_j(t) \geq 0.98 \end{cases}$$

$$\alpha_j(t) = \min\left\{\frac{t - a_j}{b_j - a_j}, 1\right\}.$$

so the token weights depend on time instead of asset balances.

In the new implementation, the model for YAMM has been changed in multiple aspects. For example,

- in the original implementation, weights for assets other than cAsset is `(Balance/cBalance)**alpha` :

```
// GetNonNormalizedWeight returns non-normalized weight for asset
// weight for cAsset is equal to exchangeRate
// weight for other assets is (Balance/cBalance)**alpha
func (yc *yammPoolController) GetNonNormalizedWeight(ctx sdk.Context, token
types.PoolToken) (weight sdk.Dec, err error) {
    return yc.computeNonNormalizedWeight(ctx, token, nil, true)
}

// computeNonNormalizedWeight returns non-normalized weight for asset given the
CBalance.
// If you want the CBalance to be read from context  use GetNonNormalizedWeight
or pass nil as balance.
// weight for cAsset is equal to exchangeRate
// weight for other assets is (Balance/cBalance)**alpha
func (yc *yammPoolController) computeNonNormalizedWeight(ctx sdk.Context, token
types.PoolToken, cBalancePtr *sdk.Dec,
    ...
    assetBalance := actualBalance.Add(virtualBalance)
    base := assetBalance.Quo(cBalance)
    alpha, err := yc.computeAlpha(ctx, token)
    if err != nil {
      return sdk.Dec{}, err
    }

    return types.PowNonNegUp(base, alpha)
}
```

- in the new implementation, weights for assets other than cAsset is calculated by `K` , which depends on time instead of asset balances:

```go
// GetNonNormalizedWeight returns non-normalized weight for asset
// weight for cAsset is equal to exchangeRate*lambda
// weight for other assets is K
func (yc *yammPoolController) GetNonNormalizedWeight(ctx sdk.Context, token
types.PoolToken) (weight sdk.Dec, err error) {
    if yc.isCAsset(token) {
      exchangeRate, err := yc.pool.GetExchangeRate(ctx)
      if err != nil {
          return sdk.Dec{}, err
      }
      lambda := yc.pool.keeper.YammLambda(ctx, yc.pool.data.Id)
      return exchangeRate.Mul(lambda), nil
    }

    return yc.computeK(ctx, token.Denom)
}
```

```go
// computeK computes the K scaler for the pool.
// k = 1/(1-alpha)
func (yc *yammPoolController) computeK(ctx sdk.Context, denom string) (sdk.Dec,
error) {
    alpha, err := yc.computeAlpha(ctx, denom)
    if err != nil {
      return sdk.Dec{}, err
    }

    return sdk.OneDec().Quo(sdk.OneDec().Sub(alpha)), nil
}

// computeAlpha computes alpha for an asset
//
//                 _                                              _
//                |    blockTime - maturityStart                      |
//    alpha = min | ----------------------------------- , maxAlpha    |
//                |_    maturityExpiry - maturityStart            _|
func (yc *yammPoolController) computeAlpha(ctx sdk.Context, denom string)
(sdk.Dec, error) {
    ...
```

As a result, the described scenario will not take place.

BRI-02 | FAILURE OF `HandleIBCTransferRecv()` DUE TO INCORRECT IBC DENOM

Category	Severity	Location	Status
Logical Issue	● Major	bridge_sweep.go (374cad8): 169	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_sweep.go`
- `x/icstaking/types/host_chain.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

Incorrect construction of the IBC denom will lead to the failure to sweep process.

The function `HandleIBCTransferRecv()` is intended to handle received IBC messages for sweep memos and handles undelegation reception, in which the IBC denom will be checked:

x/icstaking/keeper/bridge_sweep.go

```
 168
// ignore the transfer if the token denom is not equal to the expected denom
 169      if data.Denom != hostChain.IbcDenom(transferAccount) {
 170          return nil
 171      }
```

However, the creation of the IBC denom is based on the `transferAccount`, which is supposed to be the `transferChannel` (i.e., packet.DestinationChannel).

x/icstaking/types/host_chain.go

```go
// IbcDenom returns the ibc denomination of the host chain base denom
func (hostChain HostChain) IbcDenom(channel string) string {
    // FIXME store map in host chain?
    var c TransferChannel
    for _, transferChannel := range hostChain.TransferChannels {
        if transferChannel.Id == channel {
            c = transferChannel
            break
        }
    }

    denom := hostChain.BaseDenom
    if strings.TrimSpace(c.WrappedDenom) != "" {
        denom = c.WrappedDenom
    }

    return transfertypes.DenomTrace{
        Path:      fmt.Sprintf("%s/%s", transfertypes.PortID, channel),
        BaseDenom: denom,
    }.IBCDenom()
}
```

In this case, it never matches the `data.Denom` . As a result, the sweep process could never be executed successfully.

Recommendation

Recommend using `packet.DestinationChannel` to create the IBC denom.

Alleviation

[Pryzm Team - 09/13/2023] :

The team resolved the finding by correcting the denom to the host chain's base denom since the packet data contains the token's raw denom.

```go
// ignore the transfer if the token denom is not equal to the expected denom
    if data.Denom != hostChain.BaseDenom {
        return nil
    }
```

The change is reflected in the commit `dd5ae09121af41bc745ab8ce6d4c8a27624fb230` .

GEN-01 | UNEXPORTED EXPIRING TOKEN LIST MAY LEAD TO USERS' ASSET LOSS

Category	Severity	Location	Status
Coding Issue	● Major	genesis.go (374cad8): 130	● Resolved

Description

Files:

- `x/amm/keeper/genesis.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

In the AMM module, `ExportGenesis` does not export the expiring pool token list to the genesis state. It will remove all virtual balances after restarting the chain from an exported genesis state and thus lead to liquidity providers' asset loss.

For example, suppose there is an expiring pool token, a virtual balance will be introduced to incentive people to purchase this token. However, if the chain restarts from a genesis state missing the expiring pool token information, the virtual balance will become zero. As a result, the exchange rate of the token will be changed a lot after importing the incorrect genesis state, and people will use this imbalanced exchange rate to purchase tokens with low costs.

Proof of Concept

In the following unit test, the original genesis state has one expiring pool token, while the exported genesis state has zero expiring pool token.

```go
func (s *keeperTestSuite) TestGenesisInconsistency() {
    genesisState := types.GenesisState{
        Params: types.DefaultParams(),
        PoolList: []types.GenesisPoolData{
            {
                Pool: types.Pool{
                    Id:           0,
                    Creator:      sample.AccAddress(),
                    PoolType:     types.WeightedPoolType,
                    SwapFeeRatio: sdk.MustNewDecFromStr("0.1"),
                    Name:         "abcd",
                },
                TotalLpTokenSupply: sdk.ZeroInt(),
                PoolTokenList: []types.PoolToken{
                    {
                        PoolId:  0,
                        Denom:   "denom0",
                        Balance: sdk.ZeroInt(),
                    },
                    {
                        PoolId:  0,
                        Denom:   "denom1",
                        Balance: sdk.ZeroInt(),
                    },
                    {
                        PoolId:  0,
                        Denom:   "denom2",
                        Balance: sdk.ZeroInt(),
                    },
                },
            },
        },
        WeightedPoolPropertiesList: []types.WeightedPoolProperties{
            {
                PoolId: 0,
                WeightUpdateTiming: types.WeightUpdateTiming{
                    StartUnixMillis: time.Now().UTC().UnixMilli(),
                    EndUnixMillis:   time.Now().UTC().UnixMilli(),
                },
                TokenList: []types.WeightedToken{
                    {
                        PoolId:                0,
                        Denom:                 "denom0",
                        NormalizedStartWeight: sdk.MustNewDecFromStr("0.1"),
                        NormalizedEndWeight:   sdk.MustNewDecFromStr("0.1"),
                    },
                    {
                        PoolId:                0,
                        Denom:                  "denom1",
                        NormalizedStartWeight: sdk.MustNewDecFromStr("0.2"),
                        NormalizedEndWeight:   sdk.MustNewDecFromStr("0.2"),
                    },
                    {
                        PoolId:                0,
                        Denom:                 "denom2",
                        NormalizedStartWeight: sdk.MustNewDecFromStr("0.7"),
                        NormalizedEndWeight:   sdk.MustNewDecFromStr("0.7"),
                    },
                },
            },
        },
        ExpiringPoolTokenList: []types.VirtualBalancePoolToken{
            {
                PoolId:               0,
                Denom:                "denom2",
                TargetVirtualBalance: sdk.NewInt(int64(100)),
                StartUnixMillis:      time.Now().Add(-1 * time.Hour).UnixMilli(),
                EndUnixMillis:        time.Now().Add(24 * 7 *
time.Hour).UnixMilli(),
            },
        },
    }

    s.Require().NoError(genesisState.Validate())
    keeper.InitGenesis(s.ctx, *s.ammKeeper, genesisState)
    got := keeper.ExportGenesis(s.ctx, *s.ammKeeper)
    s.Require().NotNil(got)

    // original expiring pool token list has 1 element, while the exported one has 0
element
    s.Require().Len(genesisState.ExpiringPoolTokenList, 1)
    s.Require().Len(got.ExpiringPoolTokenList, 0)
}
```

Results:

```
Running tool: /usr/local/go/bin/go test -timeout 30s -testify.m
^(TestGenesisInconsistency)$ github.com/pryzm-finance/pryzm-core/x/amm/keeper

WARNING: proto: file name query.proto does not start with expected testdata/; please
make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name testdata.proto does not start with expected testdata/;
please make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name tx.proto does not start with expected testdata/; please
make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name unknonwnproto.proto does not start with expected
testdata/; please make sure your folder structure matches the proto files fully-
qualified names
PASS
ok      github.com/pryzm-finance/pryzm-core/x/amm/keeper    1.310s
```

Recommendation

Recommend exporting the expiring pool token list to the genesis state.

Alleviation

[Pryzm Team - 09/18/2023] :

The team heeded the advice and resolved the issue in the commit `1067960446fef41b4729ad1e0fd0269ec517f98a` .

KEA-02 | POTENTIAL FAILURE TO ZERO IMPACT JOIN THE YAMM POOL DUE TO THE EXPIRING OR EXPIRED PASSET

Category	Severity	Location	Status
Logical Issue	● Major	x/amm/keeper/pools/base_weighted_pool.go (pryzm-core-17e20c2): 367; x/amm/keeper/zero_impact_join_yamm.go (pryzm-core-17e20c2): 34, 96	● Resolved

Description

Files:

- `x/amm/keeper/zero_impact_join_yamm.go`
- `x/amm/keeper/pools/base_weighted_pool.go`

Commit:

- `17e20c2b046a1b389630270bfc10a1079ea0a177`

The function `executeOrQueryZeroImpactJoinYamm` implemented a zero-impact join feature that enables users to join a YAMM pool solely with cASSETs, while minimizing the price impact. So a portion of the cASSET will be refactored to pASSET for each maturity, and then using the remaining cASSET and pASSETs to join the pool. However, there is a validation in the function `JoinGivenExactTokensIn` to prevent the expiring or expired asset from joining the pool,

```
365  func (bw *baseWeightedPool) JoinGivenExactTokensIn(ctx sdk.Context,
tokensSortedByDenom []types.TokenAmount, lptSupply sdkmath.Int) (lpOut sdkmath.Int,
366      protocolFees, swapFees []sdkmath.Int, lpSwapFee sdkmath.Int, err error) {
367      err = bw.controller.ValidateJoinExactTokens(ctx, tokensSortedByDenom)
368      if err != nil {
369          return lpOut, protocolFees, swapFees, lpSwapFee, err
370      }
```

```go
// ValidateJoinExactTokens prevents joins if amount for expiring/introducing tokens is not zero

func (yc *yammPoolController) ValidateJoinExactTokens(ctx sdk.Context, tokens []types.TokenAmount) error {
    for _, ta := range tokens {
        // only check for non-zero amounts
        if ta.Amount.IsZero() {
            continue
        }

        if yc.isTokenExpiringOrExpired(ctx, ta.Token) {
            return sdkerrors.Wrapf(types.ErrInvalidJoin, "cannot join exact amount for expiring token %s", ta.Token.Denom)
        }
    }
    return nil
}
```

Hence, if any pASSET of all refactored pASSETs is expiring or expired, the zero-impact join will eventually fail.

Proof of Concept

```
func (s *keeperTestSuite) TestMsgServerZeroImpactJoinYammWithExpirePAsset() {

    s.setZeroProtocolFeeParams()
    pool, asset, _, maturities := s.createAndInitializeTwoPAssetYamm(0)

    // set time after expiration of the expiring token(pAsset1 : expire time
2024.1.1)
    s.ctx = s.ctx.WithBlockTime(time.Date(2024, 1, 2, 0, 0, 0, 0, time.UTC))
    keeper.BeginBlocker(s.ctx, *s.ammKeeper)

    s.assetsKeeper.EXPECT().
        GetRefractableAssetByTokenDenom(s.ctx, asset.TokenDenom).
        Return(asset, true).AnyTimes()

    // .                c                    |    p1                   |    p2
    // token balances: 70000e6               |  80000e6                |  82000e6
    // exchange rate = 1.2
    // refract fee = 0.01
    // effective exchange rate = 1.2 * (1 - 0.01) = 1.188
    // c*effective er = 70000e6 * 1.188 = 83160e6
    // total p = 83160e6 + 80000e6 + 82000e6 = 245160e6
    // refract for each maturity = 326317, 334475
    s.assetsKeeper.EXPECT().
        RefractorRefractFeeRatio(s.ctx, asset.Id).
        Return(sdk.MustNewDecFromStr("0.01")).
        AnyTimes()

    refractorAction1 :=
refractormodulekeeper.NewRefractorAction(refractormodulekeeper.RefractActionType,
        asset, maturities[0], sdk.NewInt(3265), sdk.NewInt(323053),
        sdk.NewInt(326318), sdk.NewInt(387663), sdk.NewInt(387663))
    s.refractorKeeper.EXPECT().
        ComputeRefract(s.ctx, sdk.NewCoin(asset.TokenDenom, sdk.NewInt(326318)),
            maturities[0].Symbol).
        Return(refractorAction1, nil).
        Times(1)
    refractorAction2 :=
refractormodulekeeper.NewRefractorAction(refractormodulekeeper.RefractActionType,
        asset, maturities[1], sdk.NewInt(3345), sdk.NewInt(331130),
        sdk.NewInt(334475), sdk.NewInt(397356), sdk.NewInt(397356))
    s.refractorKeeper.EXPECT().
        ComputeRefract(s.ctx, sdk.NewCoin(asset.TokenDenom, sdk.NewInt(334475)),
            maturities[1].Symbol).
        Return(refractorAction2, nil).
        Times(1)

    addrStr := sample.AccAddress()
    addr := sdk.MustAccAddressFromBech32(addrStr)
    agg := refractormodulekeeper.NewAggregatedRefractorAction(addr)
    s.Require().NoError(agg.Append(refractorAction1))
    s.Require().NoError(agg.Append(refractorAction2))
    s.refractorKeeper.EXPECT().
        ExecuteAggregatedAction(s.MatchContext(), agg).
        Return(nil).Times(1)

    // execute message to make sure the message changes the db
    // and simulation responses are equal to actual message responses
    s.bankKeeper.EXPECT().
        BlockedAddr(addr).
        Return(false)
    s.setupMocksAllowAnyTransferMintAndBurn()

    messageResponse, err := s.msgServer.ZeroImpactJoinYamm(s.ctx,
&types.MsgZeroImpactJoinYamm{
        Creator:   addrStr,
        CAmountIn: sdk.NewInt64Coin(asset.TokenDenom, 1000000),
    })
    s.Require().NoError(err)
    // simulation output should be equal to actual message execution outputs
    s.Require().Equal(&types.MsgZeroImpactJoinYammResponse{
        LptOut: sdk.NewInt64Coin(pool.GetLpDenom(), 339205),
        YOut: sdk.NewCoins(sdk.NewInt64Coin(assets.YDenom(asset.Id,
maturities[0].Symbol), 387663),
            sdk.NewInt64Coin(assets.YDenom(asset.Id, maturities[1].Symbol),
397356)),
        RefractFee:      sdk.NewInt64Coin(asset.TokenDenom, 6610),
        JoinProtocolFee: sdk.NewCoins(),
        SwapFee:         sdk.NewCoins(),
    }, messageResponse)
}
```

Output:

```
=== RUN   TestKeeperTestSuite
=== RUN   TestKeeperTestSuite/TestMsgServerZeroImpactJoinYammWithExpirePAsset
    msg_server_zero_impact_join_yamm_test.go:164:
            Error Trace:    /Users/certik/go_audit_project/pryzm-finance/pryzm-core-
1004/x/amm/keeper/msg_server_zero_impact_join_yamm_test.go:164
            Error:          Received unexpected error:

                            github.com/pryzm-finance/pryzm-core/x/amm/keeper/pools.
(*yammPoolController).ValidateJoinExactTokens
```

Recommendation

We recommend considering the scenario where the pASSET is either expiring or has already expired.

Alleviation

[Pryzm Team - 10/16/2023] :

The team resolved the finding by excluding the expiring or expired PAsset to fix the issue in the commit `07d035e82c02369585f33b0d69cdfbc6c98ecb17` .

KEE-04 | PENDINGCAMOUNT, PENDINGAMOUNT, AND RECEIVEDAMOUNT ARE NOT UPDATED AFTER HANDLING UNDELEGATION RECEPTION

Category	Severity	Location	Status
Logical Issue	● Major	bridge_sweep.go (374cad8): 122, 180; msg_server_redeem_unstaked.go (374cad8): 45~46	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_sweep.go`
- `x/icstaking/keeper/msg_server_redeem_unstaked.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The properties of `ChannelUndelegation` mentioned below have a direct impact on the undelegation state and should be updated accordingly after the undelegation reception:

- PendingCAmount
- PendingAmount
- ReceivedAmount
- Swept

Among them, the value of `ReceivedAmount` plays a crucial role in calculating the redemption rate of the undelegation, and eventually affects the asset amount that will be redeemed to users.

- `x/icstaking/keeper/msg_server_redeem_unstaked.go`

```
45
// calculate the amount to be redeemed to user, based on the redemption rate of the
undelegation

46    redemptionRate := sdk.NewDecFromInt(undelegation.ReceivedAmount).QuoInt(
undelegation.TotalCAmount)
```

Hence, users are unable to get any undelegated underlying assets even if they provided all amounts of uAsset.

Scenario

To reproduce the error, please follow the steps:

1. Take a specific quantity of underlying assets from a user's account, generating an equivalent amount of `cAsset` for the user
2. Wait for the delegation operation to be executed at the start of the subsequent block
3. The user initiates an unstaking of their `cAsset`.
4. Wait for the undelegation operation to be executed at the start of the next block
5. PRYZM updates the host chain state upon receiving information from the Oracle, modifying channel undelegations, and marking `received` as true.
6. However, in Step 5, the value of undelegation.ReceivedAmount does not undergo any modification
7. Since the `undelegation.ReceivedAmount` remains stagnant at zero without any updates, users face an inability to redeem the underlying asset

Recommendation

It is important to ensure that these properties are correctly updated and reflect the relevant state changes after the undelegation reception. We recommend reviewing the logic again and updating these properties accordingly after handling undelegation reception.

Alleviation

[Pryzm Team - 09/15/2023] :

The team heeded the advice and resolved this issue in the commit `cc4da59bae539c2a5c15090491aa7fda4da4987c`.

```
    for channel, sweep := range data.ChannelSweeps {
        for _, epoch := range sweep.Epochs {
            channelUndelegation, found := b.keeper.GetChannelUndelegation(ctx, replyData.HostChainId, epoch, channel)
            if !found {
                continue
            }
            channelUndelegation.Swept = true
            b.keeper.setChannelUndelegation(ctx, channelUndelegation)
        }
    }
```

```
    channelUndelegation.ReceivedAmount = channelUndelegation.ReceivedAmount.Add(amount)
    channelUndelegation.PendingAmount = sdk.ZeroInt()
    channelUndelegation.PendingCAmount = sdk.ZeroInt()
```

KEE-05 | POTENTIAL PANICS WHEN FETCHING NIL VALIDATOR FROM `weightDiff`

Category	Severity	Location	Status
Volatile Code	● Major	bridge_delegate.go (374cad8): 155~167; bridge_redelegate.go (374cad8): 124~134; bridge_undelegate.go (374cad8): 188~198	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_delegate.go`
- `x/icstaking/keeper/bridge_redelegate.go`
- `x/icstaking/keeper/bridge_undelegate.go`
- `x/icstaking/keeper/host_chain.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The nil validator dereference will be triggered to block the operations of delegation, undelegation, and redelegation if there are any missing validators from the host chain that are registered on Pryzm, for example, if a new validator has been registered but the host chain state has not been updated. More precisely, the function `UpdateHostChain()` is intended to update the host chain registration information via governance, either with the new parameters or new validators.

x/icstaking/keeper/host_chain.go

```go
func (k Keeper) UpdateHostChain(ctx sdk.Context, hostChainID string, validators types.Validators, params *types.StakingParams) error {
    hostChain, found := k.GetHostChain(ctx, hostChainID)
    if !found {
        return sdkerrors.Wrapf(types.ErrHostChainNotFound, hostChainID)
    }

    if params != nil {
        hostChain.Params = *params
    }

    if len(validators) != 0 {
        // descending sort of validators by their weight
        validators.SortByWeight(true)

        // create a map of new validators list
        newValMap := make(map[string]bool)
        for _, v := range validators {
            newValMap[v.Address] = true
        }

// add the old validators which are not included in the new validator set with weight 0

        for _, oldVal := range hostChain.Validators {
            if !newValMap[oldVal.Address] {
                validators = append(validators, types.Validator{Address: oldVal.Address, Weight: sdk.ZeroDec()})
            }
        }
        hostChain.Validators = validators
    }

    k.setHostChain(ctx, hostChain)
    return nil
}
```

However, the validators in the host chain state has not been updated. In case that new validator is introduced in host chain, then it can not be found in the host chains state.

According to the construction of slice `weightDiff` , if some validator is not found in the `hostChainState` from the host chain, the corresponding position in the slice will be nil.

```
    // create a mapping of validators to the diff of their current weight to their
expected weight
    weightDiff := make([]types.Validator, len(hostChain.Validators))
    for i, validator := range hostChain.Validators {
        valInfo, found := hostChainState.Validators[validator.Address]
        if !found {
            continue
        }
        actualWeight :=
sdk.NewDecFromInt(valInfo.DelegatedAmount).QuoInt(totalDelegation)
        weightDiff[i] = types.Validator{
            Address: validator.Address,
            Weight:  actualWeight.Sub(expectedWeights[validator.Address]),
        }
    }
```

In this case, fetching the nil validator with the invocation of methods `LT()`, `GT()`, or `Abs()` could possibly lead to panic.

```
    // ascending sort
    sort.SliceStable(weightDiff, func(i, j int) bool {
        return weightDiff[i].Weight.LT(weightDiff[j].Weight)
    })

    // create a mapping of validator to their share of current delegation
    delegationMap := make(map[string]math.Int)
    remainingDelegation := totalDeposit
    for i := 0; i < len(weightDiff) && !remainingDelegation.IsZero(); i++ {
        diffAmount :=
weightDiff[i].Weight.Abs().MulInt(totalDelegation).TruncateInt()
        delegationAmount := sdk.MinInt(diffAmount, remainingDelegation)

        delegationMap[weightDiff[i].Address] = delegationAmount
        remainingDelegation = remainingDelegation.Sub(delegationAmount)
    }
```

Proof of Concept

The following unit test is used to reproduce the panics:

```
func Test_nilValidator(t *testing.T) {
    weightDiff := make([]types.Validator, 2)
    absWeight := weightDiff[0].Weight.Abs()
    fmt.Printf("The absolute weight is %f\n", absWeight)
}
```

Result:

```
=== RUN   Test_nilValidator
--- FAIL: Test_nilValidator (0.00s)
panic: runtime error: invalid memory address or nil pointer dereference [recovered]
    panic: runtime error: invalid memory address or nil pointer dereference
```

Recommendation

Recommend sanitizing the validator that can not be found in the host chain state.

Alleviation

[Pryzm Team - 09/13/2023] :

The team heeded the advice and refactored the logic of calculating the amount of delegation and undelegation to fix the issue in the commit `074a272bb61adb89b9a040ff9dadbc2634e572a8` .

KEE-06 | INCORRECT DISTRIBUTION OF DELEGATION AND UNDELEGATION AMOUNT AS THE LAST VALIDATOR GETS ENTIRE `remainingUndelegation` AND `remainingDelegation`

Category	Severity	Location	Status
Incorrect Calculation	● Major	bridge_delegate.go (374cad8): 192~199; bridge_undelegate.go (374cad8): 223~230	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_delegate.go`
- `x/icstaking/keeper/bridge_undelegate.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The incorrect distribution of the remaining delegation and undelegation amount among the validators would possibly cause more delegation and undelegation amount to be executed, respectively.

In functions `CreateDelegationMsgs()` and `createUndelegationMsgs()` , the remaining delegation and undelegation amount will be distributed to the validators based on their expected weights.

x/icstaking/keeper/bridge_delegate.go

```go
    // distribute the remaining uniformly based on each validator weight
    if !remainingDelegation.IsZero() {
        w := sdk.ZeroDec()
        for val := range delegationMap {
            w = w.Add(expectedWeights[val])
        }
        i := 0
        for val := range delegationMap {
            if i == len(delegationMap)-1 {
                delegationMap[val] = delegationMap[val].Add(remainingDelegation)
            } else {
                delegationMap[val] = delegationMap[val].Add(expectedWeights[val].Quo(w).MulInt(remainingDelegation).TruncateInt())
            }
            i++
        }
    }
```

x/icstaking/keeper/bridge_undelegate.go

```go
    if !remainingUndelegation.IsZero() {
        w := sdk.ZeroDec()
        for val := range undelegationMap {
            w = w.Add(expectedWeights[val])
        }
        i := 0
        for val := range undelegationMap {
            if i == len(undelegationMap)-1 {
                undelegationMap[val] = undelegationMap[val].Add(remainingUndelegation)
            } else {
                undelegationMap[val] = undelegationMap[val].Add(expectedWeights[val].Quo(w).MulInt(remainingUndelegation).TruncateInt())
            }
            i++
        }
    }
```

As shown in the above implementation, the previous n-1 validators get the delegation and undelegation amount proportional to their expected weights. However, the last validator in the map gets the entire `remainingDelegation` and `remainingUndelegation` , which are supposed to be the amount left over. As a result, more delegation and undelegation amount will be produced.

Proof of Concept

To demonstrate the scenario, we use the following unit test.

1. set the remaining delegation as 100;

2. set two validators 1 and 2 with expected weights, 0.9 and 0.1;
3. Validator 1 will get 100 * 0.9 = 90, and validator 2 will get 100.

```go
func Test_remainingDelegationDistribution(t *testing.T) {
    remainingDelegation := math.NewInt(100)
    val1Address := "1"
    val2Address := "2"

    delegationMap := map[string]math.Int{
        val1Address: sdk.NewInt(0),
        val2Address: sdk.NewInt(0),
    }

    expectedWeights := map[string]sdk.Dec{
        val1Address: sdk.MustNewDecFromStr("0.9"),
        val2Address: sdk.MustNewDecFromStr("0.1"),
    }

    // distribute the remaining uniformly based on each validator weight
    if !remainingDelegation.IsZero() {
        w := sdk.ZeroDec()
        for val := range delegationMap {
            w = w.Add(expectedWeights[val])
        }
        i := 0
        for val := range delegationMap {
            if i == len(delegationMap)-1 {
                delegationMap[val] = delegationMap[val].Add(remainingDelegation)
            } else {
                delegationMap[val] =
delegationMap[val].Add(expectedWeights[val].Quo(w).MulInt(remainingDelegation).Trunc
ateInt())
            }
            i++
        }
    }
    // calculate the total delegation that has been distributed
    totalDelegation := math.NewInt(0)
    for val := range delegationMap {
        totalDelegation = totalDelegation.Add(delegationMap[val])
    }
    require.NotEqual(t, totalDelegation, remainingDelegation)
    fmt.Printf("The total delegation is %s, not %s!\n", totalDelegation,
remainingDelegation)
}
```

Result:

```
=== RUN   Test_remainingDelegationDistribution
The total delegation is 190, not 100!
--- PASS: Test_remainingDelegationDistribution (0.00s)
PASS
```

Recommendation

Recommend correcting the distribution logic so that the last validator gets the left over.

Alleviation

[Pryzm Team - 09/15/2023] :

The team heeded the advice and resolved this issue in the commit `074a272bb61adb89b9a040ff9dadbc2634e572a8` .

KEP-02 POTENTIALLY UNABLE TO EXIT THE FLOW SUCCESSFULLY DUE TO THE FLOW BEING STOPPED BY THE FLOW CREATOR

Category	Severity	Location	Status
Logical Issue	● Major	x/flowtrade/keeper/flow.go (flowtrade): 102; x/flowtrade/keeper/position.go (flowtrade): 104	● Resolved

Description

Files:

- `x/flowtrade/keeper/flow.go`
- `x/flowtrade/keeper/position.go`

Commit:

- `930876154d4296a366ba2ca179c227c6663cc55b`

In the `flowtrade` module, a user can offer `token-out` tokens and lock deposits to establish a flow. Meanwhile, other users can participate in an active flow by contributing a certain amount of `token-in` tokens to purchase token-out tokens. The participants can also choose to exit and retrieve their unexchanged `token-in` tokens if the flow is not ended or stopped:

- `x/flowtrade/keeper/position.go`

```go
func (k Keeper) ExitFlow(ctx sdk.Context, flowId uint64, address sdk.AccAddress,
amount sdk.Coin) error {
    ...

    // return error if flow is not active
    switch flow.Status {
    case types.FlowStatus_ENDED:
        return types.ErrFlowEnded
    case types.FlowStatus_STOPPED:
        return types.ErrFlowStopped
    }

    ...

    // transfer the amount to the user account
    err := k.sendCoinsFromModuleToAccount(ctx, address, sdk.NewCoins(amount))
    if err != nil {
        return err
    }

    return ctx.EventManager().EmitTypedEvent(&types.EventExitFlow{
        FlowId:  flowId,
        Address: address.String(),
        Amount:  amount,
    })
}
```

However, it should be noted that the `StopFlow()` function grants the creator of the flow the authority to stop the flow at their discretion:

- `x/flowtrade/keeper/flow.go`

```go
// StopFlow stops flow and update it in the store
func (k Keeper) StopFlow(ctx sdk.Context, flowId uint64) error {
    ...

    // update to sync indices with the stopping time
    flow.UpdateDistIndex(ctx.BlockTime())
    flow.Status = types.FlowStatus_STOPPED

    ...
}
```

This action could potentially result in the loss of `token-in` tokens for users who have joined the flow and have not exited and withdrawn their unexchanged `token-in` tokens before the flow is stopped.

Proof of Concept

```
func (s *keeperTestSuite) TestJoinExitFlow() {
    tokenOutDenom := "token-out"
    tokenInDenom := "token-in"
    address := sdk.MustAccAddressFromBech32(sample.AccAddress())
    joiner1 := sdk.MustAccAddressFromBech32(sample.AccAddress())
    now := time.Now()

    params := types.DefaultParams()
    params.MinFlowDuration = time.Hour
    params.MinDurationToFlowStart = 0
    err := s.flowtradeKeeper.SetParams(s.ctx, params)
    s.Require().NoError(err)

    // create a flow with 0 dist interval
    start := now
    end := now.Add(4 * time.Hour)
    request := types.NewFlowCreationRequest(
        types.NewFlowInfo("", "", ""),
        start, end, 0,
        address,
        sdk.NewInt64Coin(tokenOutDenom, 1_000_000),
        tokenInDenom,
        end, end,
        true, false, false,
    )
    ctx := s.ctx.WithBlockTime(now.Add(-4 * time.Hour))
    s.bankKeeper.EXPECT().SendCoins(gomock.Any(), address, s.moduleAddress,
sdk.NewCoins(request.TokensOut)).Return(nil).Times(1)
    flowId, err := s.flowtradeKeeper.CreateFlow(ctx, address, request, false, nil,
sdk.ZeroDec(), sdk.ZeroDec())
    s.Require().NoError(err)

    ctx = ctx.WithBlockTime(start.Add(4 *
time.Hour)).WithEventManager(sdk.NewEventManager())
    amount := sdk.NewInt64Coin(tokenInDenom, 1_000_000)
    s.bankKeeper.EXPECT().SendCoins(gomock.Any(), joiner1, s.moduleAddress,
sdk.NewCoins(amount)).Return(nil).Times(1)
    err = s.flowtradeKeeper.JoinFlow(ctx, flowId, joiner1, amount)
    s.Require().NoError(err)

    flow, _ := s.flowtradeKeeper.GetFlow(ctx, flowId)
    flow.Status = types.FlowStatus_STOPPED     // Stop the flow
    s.flowtradeKeeper.SetFlow(ctx, flow)

    amount = sdk.NewInt64Coin(tokenInDenom, 1_000_000)
    err = s.flowtradeKeeper.ExitFlow(ctx, flowId, joiner1, amount)
    s.Require().Error(err)
    position, _ := s.flowtradeKeeper.GetPosition(ctx, flowId, joiner1.String())
    s.Require().Equal(position.TokenInBalance, sdk.NewInt(1_000_000))
}
```

Recommendation

It is important to consider the scenario mentioned and ensure that all users who have joined the flows can successfully exit the flows and withdraw their unexchanged tokens. This will help to promote fairness and provide a smooth user experience within the system.

Alleviation

[Pryzm Team - 09/15/2023] :

The team heeded the advice and removed final state validation from exit flow method to resolve this issue in the commit `1f3f76d316391b34131f6c7cb5178a349e61ade8` .

KER-02 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	● **Major**	**msg_server_remove_token_from_weighted_pool.go (374cad8): 20; msg_server_set_circuit_breakers.go (374cad8): 20; msg_server_set_initialization_allow_list.go (374cad8): 20; msg_server_set_pause_mode.go (374cad8): 13, 24; msg_server_set_yamm_configuration.go (374cad8): 20; msg_server_update_swap_fee.go (374cad8): 19; msg_server_update_weights.go (374cad8): 20**	● **Acknowledged**

Description

Files:

- `x/amm/keeper/*`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

In the AMM module, a pool's creator has authority over the following functionalities:

- Remove a token from the pool.
- Set Circuit Breakers, which are used to prevent extreme price movements of tokens, for the pool.
- Set the list of accounts that can initialize the pool.
- Set pause mode for the pool.
- Set configuration for the pool if it is a YAMM pool.
- Update the swap fee ratio for the pool.
- Update weights of tokens in the pool.

A weighted pool can be created by the `authority` account or any account if public pool creations are allowed:

```go
func (k msgServer) CreateWeightedPool(goCtx context.Context, msg *types.MsgCreateWeightedPool) (*types.MsgCreateWeightedPoolResponse, error) {
	ctx := sdk.UnwrapSDKContext(goCtx)
	if !k.AllowPublicPoolCreation(ctx) && k.authority != msg.Creator {
		return nil, sdkerrors.Wrapf(govtypes.ErrInvalidSigner, "invalid creator; expected %s, got %s", k.authority, msg.Creator)
	}
```

The `authority` account is set to `govModuleAddress` (the last argument) in app.go:

```go
	app.AmmKeeper = *ammmodulekeeper.NewKeeper(
		appCodec,
		keys[ammmoduletypes.StoreKey],
		keys[ammmoduletypes.MemStoreKey],

		app.AccountKeeper,
		app.BankKeeper,
		app.AssetsKeeper,
		app.RefractorKeeper,
		app.TreasuryKeeper,
		govModuleAddress,
	)
```

And `AllowPublicPoolCreation` as a parameter can only be updated by `gov`, which allows staking token holders to vote on proposals. Therefore, the creation of weighted pools is restricted by `gov`.

However, it should be noted that anyone can create weighted pools when `AllowPublicPoolCreation` is set `true` by `gov` and perform the aforementioned privileged operations.

Meanwhile, even though people can create weighted pools when `AllowPublicPoolCreation` is `true`, `gov` would still be able to pause the pools:

```go
func (k msgServer) SetPauseMode(goCtx context.Context, msg *types.
MsgSetPauseMode) (*types.MsgSetPauseModeResponse, error) {
	ctx := sdk.UnwrapSDKContext(goCtx)
	if k.authority == msg.Creator {
		err := k.SetGovPauseMode(ctx, msg.PoolId, msg.PauseMode)
		if err != nil {
			return nil, err
		}
```

In conclusion, pools' creations and operations are subject to regulation by `gov`, while normal users can create pools and affect pool operations only when granted permission by `gov`.

Any compromise to pools' creator accounts may allow a hacker to take advantage of this authority and manipulate the pools.

Scenario

Scenario 1

The pool creator role can update the swap fee rate without seeking consensus from users or providing them with notifications, which might lead to users paying unexpected swap fees.

Imagine a scenario where a user intends to swap a large amount of TokenA for TokenB in a pool. If the pool creator, noticing this transaction, preemptively alters the swap fee from 1% to 10% (the maximum allowed ratio), the user would incur unexpectedly high fees. Such an action by a pool creator not only undermines trust but also poses a significant financial risk to users, albeit indirectly.

Scenario 2

The pool creator role can update token weights in the pool without seeking consensus from users or providing them with notifications.

These token weights play a crucial role in the pool's constant product equation, represented as $\Pi_i B_i^{w_i} = C$. Adjusting these weights alters the value of tokens in the pool. Consequently, this can impact the outcomes of token swaps, as well as the addition or removal of liquidity. As a result, changes to the token weights can indirectly influence the value of users' funds in the pool.

[Note - 11/29/2023]: This scenario is connected to the finding **WEI-02**, which has been resolved in PR#236. As a result, the token weights will always be updated **gradually**.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND

- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Pryzm Team - 09/15/2023]: Currently, we have two types of pools: YAMM pools and normal weighted pools (balancer pools). The first kind of pool is not allowed to be created by non-gov addresses. The weighted pools can be created by non-gov address only if gov has decided to allow that. Therefore the core functionality and main pools are always created by governance and we are not relying on user-created pools for core functionalities. However, if gov decide to allow public creation and some non-gov creates a pool, they are in charge of configuring the pool, e.g. changing the weights any time they want to. even in this situation, we still have a gov feature to pause a pool and set the pool to recovery mode so liquidity providers can withdraw their liquidity in an emergency. Please note that the pool configurations can change the prices in the pool but they do not allow for taking liquidity out of the pool through the owner account. It is up to users to decide which pools they can rely on and which pools are owned by suspicious accounts. For example, you can only trust pools created by gov or multi-sig accounts.

[CertiK - 10/18/2023]: The team agrees that users should be careful about pool creators when interacting with AMM pools. The described centralization risk is not eliminated. CertiK encourages the project team to introduce proper mechanisms to mitigate this risk in the future.

[CertiK - 11/29/2023]: In PR#236, a minimum period for updating token weights is established, necessitating a gradual approach to the token weight update process.

X0C-03 POTENTIAL CONSENSUS FAILURE BY NON-DETERMINISM OF MAP ITERATION

Category	Severity	Location	Status
Volatile Code, Denial of Service	● Major	x/amm/keeper/order_execution.go (pryzm-core-0c34472): 387; x/amm/keeper/vault_batch_swap.go (pryzm-core-0c34472): 366, 379, 452, 464, 479, 493; x/icstaking/keeper/bridge_delegate.go (pryzm-core-0c34472): 155, 173; x/icstaking/keeper/bridge_redelegate.go (pryzm-core-0c34472): 165, 180; x/icstaking/keeper/bridge_sweep.go (pryzm-core-0c34472): 47, 101; x/icstaking/keeper/bridge_undelegate.go (pryzm-core-0c34472): 44, 196, 213, 223; x/incentives/types/bond.go (pryzm-core-0c34472): 33, 55; x/incentives/types/pool.go (pryzm-core-0c34472): 38, 102, 187; x/pgov/keeper/tally.go (pryzm-core-0c34472): 66; x/refractor/keeper/keeper_action.go (pryzm-core-0c34472): 16	● Resolved

Description

Files:

- `x/amm/keeper/order_execution.go`
- `x/amm/keeper/vault_batch_swap.go`
- `x/icstaking/keeper/bridge_delegate.go`
- `x/icstaking/keeper/bridge_redelegate.go`
- `x/icstaking/keeper/bridge_sweep.go`
- `x/icstaking/keeper/bridge_undelegate.go`
- `x/incentives/types/bond.go`
- `x/incentives/types/pool.go`
- `x/pgov/keeper/tally.go`
- `x/refractor/keeper/keeper_action.go`

Commit:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The map iteration of Go is non-deterministic that each iteration of the same map will create a different order in an unpredictable manner. To ensure that the results of chain state updates are consistent among validators, it's important to always keep the iteration in the same order, which could be achieved via sorting of the keys.

In particular, the following scenarios could lead to inconsistent results of execution:

- early exit from the map iteration via break or return
- the result of each key-value pair is appended to a slice

In both cases, the result after every execution could possibly be different. If the execution result impacts the consensus or storage, then the validators may not be able to agree on the chain state, which could cause the consensus failure and chain halt.

Reference:

- https://go.dev/blog/maps
- https://github.com/cosmos/cosmos-sdk/issues/13039

Proof of Concept

To demonstrate the non-determinism of the map iteration in Go, we provide the simple test case:

```go
package main

import (
    "fmt"
)

func main() {
    myMap := map[int]string{
        1: "One",
        2: "Two",
        3: "Three",
        4: "Four",
        5: "Five",
    }

    fmt.Println("Original map:")
    printMap(myMap)

    fmt.Println("\nIterating over the map again:")
    printMap(myMap)
}

func printMap(m map[int]string) {
    for key, value := range m {
        fmt.Printf("%d: %s\n", key, value)
    }
}
```

Result:

```
Original map:
3: Three
4: Four
5: Five
1: One
2: Two

Iterating over the map again:
5: Five
1: One
2: Two
3: Three
4: Four
```

Recommendation

Recommend utilizing the sorted keys for the map iteration to ensure the determinism of every execution among the validators.

Alleviation

[Pryzm Team - 10/16/2023] :

The team heeded the advice and resolved this issue in the commit `5d65b265ab2eded4acd760f3c766c2059b24784a` .

APP-01 POTENTIAL DOS ATTACK AS CUSTOM MODULE ACCOUNTS ARE NOT INITIALIZED

Category	Severity	Location	Status
Denial of Service	● Medium	app/app.go (pryzm-core-17e20c2): 274	● Resolved

Description

Files:

- `app.go`

Commits:

- `17e20c2b046a1b389630270bfc10a1079ea0a177`

Most custom module accounts in Pryzm are not initialized during the genesis stage, which could allow malicious users to initialize the these address as base accounts beforehand. It leads to the failure of operations in these modules because the assertion of these module address as module accounts will fail.

For example, in the `incentives` module, the `tx_bond` invokes the `Bond()`:

x/incentives/keeper/msg_server_bond.go

```go
12  func (k msgServer) Bond(goCtx context.Context, msg *types.MsgBond) (*types.MsgBondResponse, error) {
13      ctx := sdk.UnwrapSDKContext(goCtx)
14
15      creator, err := sdk.AccAddressFromBech32(msg.Creator)
16      if err != nil {
17          return nil, err
18      }
19
20      bond, err := k.Keeper.BondAmount(ctx, creator, msg.Amount)
21      if err != nil {
22          return nil, err
23      }
24
25      return &types.MsgBondResponse{
26          Bond: bond,
27      }, nil
28  }
```

which calls the function `BondAmount()` from the keeper (in line 20):

x/incentives/keeper/bond.go

```go
func (k Keeper) BondAmount(ctx sdk.Context, address sdk.AccAddress, amount sdk.
Coin) (bond typesv1.Bond, err error) {
    addressStr := address.String()
    bond, err = k.bondAmountWithoutTransfer(ctx, addressStr, amount)
    if err != nil {
        return bond, err
    }

    err = k.bankKeeper.SendCoinsFromAccountToModule(ctx, address, types.
ModuleName,
        sdk.NewCoins(amount))
    if err != nil {
        return typesv1.Bond{}, err
    }

    return bond, ctx.EventManager().EmitTypedEvent(&typesv1.EventBond{
        Address: addressStr,
        Amount:  amount,
    })
}
```

The function `BondAmount()` will invoke the `SendCoinsFromAccountToModule()` from bank module to transfer the token from the creator of the bond to the `incentives` module account.

x/bank/keeper/keeper.go

```go
func (k BaseKeeper) SendCoinsFromAccountToModule(
    ctx sdk.Context, senderAddr sdk.AccAddress, recipientModule string, amt sdk
.Coins,
) error {
    recipientAcc := k.ak.GetModuleAccount(ctx, recipientModule)
    if recipientAcc == nil {
        panic(sdkerrors.Wrapf(sdkerrors.ErrUnknownAddress,
"module account %s does not exist", recipientModule))
    }

    return k.SendCoins(ctx, senderAddr, recipientAcc.GetAddress(), amt)
}
```

However, the function `GetModuleAccount()` checks if the account is indeed a **module account**:

x/auth/keeper/keeper.go

```go
func (ak AccountKeeper) GetModuleAccount(ctx sdk.Context, moduleName string)
 types.ModuleAccountI {
    acc, _ := ak.GetModuleAccountAndPermissions(ctx, moduleName)
    return acc
}
```

Otherwise, it panics if it is a **base account** in the function `GetModuleAccountAndPermissions()`:

x/auth/keeper/keeper.go

```go
func (ak AccountKeeper) GetModuleAccountAndPermissions(ctx sdk.Context,
moduleName string) (types.ModuleAccountI, []string) {
    addr, perms := ak.GetModuleAddressAndPermissions(moduleName)
    if addr == nil {
        return nil, []string{}
    }

    acc := ak.GetAccount(ctx, addr)
    if acc != nil {
        macc, ok := acc.(types.ModuleAccountI)
        if !ok {
            panic("account is not a module account")
        }
        return macc, perms
    }

    // create a new module account
    macc := types.NewEmptyModuleAccount(moduleName, perms...)
    maccI := (ak.NewAccount(ctx, macc)).(types.ModuleAccountI)
// set the account number
    ak.SetModuleAccount(ctx, maccI)

    return maccI, perms
}
```

Since the `incentives` has not been initialized as a module account during the launch, a malicious user could generates the `incentives` address deterministically with function `NewModuleAddress()`:

x/auth/types/account.go

```go
func NewModuleAddress(name string) sdk.AccAddress {
    return sdk.AccAddress(crypto.AddressHash([]byte(name)))
}
```

After that, the malicious user initializes the `incentives` as a base account by creating a periodic vesting for it (Notice that a direct bank send could also create a base account for the `incentives` address, but it will fail by the configuration that these modules are not allowed to receive external coins).

Once the bond operation is performed, it asserts that the passed account is a module account. As a result, it will lead to panics, which basically disables the bond operations as well as other operations that involve the module account assertion.

Scenario

Considering the following scenario to the bond operation in `incentives` module:

1. Suppose that any operations in the `incentives` module has not been utilized so that the `incentives` module account has not been created;
2. A malicious user, Bob generates the `incentives` address with the above function `NewModuleAddress()` ;
3. Bob creates a periodic vesting account with this `incentives` address so that the `incentives` becomes a base account;
4. Any bond operations will be disabled as the `incentives` has already been initialized as base account that leads to the assertion failure.
5. The similar scenario also applies to other modules.

Recommendation

Recommend initializing all custom module addresses as the module accounts in function `InitGenesis()` .

Alleviation

[Pryzm Team - 03/28/2024]: The team heeded the advice and resolved the finding by initializing the module accounts in the commits `4cdb204d459cd28ae1942a86fc795debae5b7ec3` and `91ca8a06ff9ace5691aa1f4c5c267b68870dcadc` .

BRD-03 | INCORRECT CALCULATION LOGIC ON `totalDelegation`

Category	Severity	Location	Status
Logical Issue	● Medium	bridge_undelegate.go (374cad8): 185, 209~215	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_undelegate.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

In the `createUndelegationMsgs` function, the calculation of the `undelegation` amount for each validator is performed, and the corresponding `undelegation` messages are generated. However, there is an issue at line 185 where the `totalUndelegation` value is mistakenly added to the `totalDelegation` instead of being subtracted from it. As a result, a negative difference weight is generated at line 196, and this can potentially trigger redundant calculation logic from lines 217 to 230.

The auditing team understands that regardless of whether `totalUndelegation` is added to `totalDelegation` or subtracted from it, each validator will ultimately have nearly the same amount of `undelegation` .

```go
func (k Keeper) createUndelegationMsgs(ctx sdk.Context, hostChain types.HostChain, hostChainState types.HostChainState, totalUndelegation math.Int) ([]*stakingtypes.MsgUndelegate, error) {
    totalDelegation := sdk.ZeroInt()
    expectedWeights := make(map[string]sdk.Dec, len(hostChain.Validators))
    for _, v := range hostChain.Validators {
        expectedWeights[v.Address] = v.Weight
        valInfo, found := hostChainState.Validators[v.Address]
        if !found {
            continue
        }
        totalDelegation = totalDelegation.Add(valInfo.DelegatedAmount)
    }

// Subtract undelegating amount from total delegation so the weight diffs are calculated considering new undelegation

    totalDelegation = totalDelegation.Add(totalUndelegation)

    weightDiff := make([]types.Validator, len(hostChain.Validators))
    for i, v := range hostChain.Validators {
        valInfo, found := hostChainState.Validators[v.Address]
        if !found {
            continue
        }
        actualWeight := sdk.NewDecFromInt(valInfo.DelegatedAmount).QuoInt(totalDelegation)
        weightDiff[i] = types.Validator{
            Address: v.Address,
            Weight:  actualWeight.Sub(expectedWeights[v.Address]),
        }
    }

    // descending sort
    sort.SliceStable(weightDiff, func(i, j int) bool {
        return weightDiff[i].Weight.GT(weightDiff[j].Weight)
    })

    maxUndelegateMsgs := k.MaxUndelegationMsgs(ctx, hostChain.GetID())

    undelegationMap := make(map[string]math.Int)
    remainingUndelegation := totalUndelegation
    for i := int32(0); i < maxUndelegateMsgs && i < int32(len(weightDiff)) && !remainingUndelegation.IsZero(); i++ {
        diffAmount := weightDiff[i].Weight.MulInt(totalDelegation).TruncateInt()
        delegationAmount := sdk.MinInt(diffAmount, remainingUndelegation)

        undelegationMap[weightDiff[i].Address] = delegationAmount
        remainingUndelegation = remainingUndelegation.Sub(delegationAmount)
    }

    if !remainingUndelegation.IsZero() {
        w := sdk.ZeroDec()
        for val := range undelegationMap {
            w = w.Add(expectedWeights[val])
        }
        i := 0
        for val := range undelegationMap {
            if i == len(undelegationMap)-1 {
                undelegationMap[val] = undelegationMap[val].Add(
remainingUndelegation)
            } else {
                undelegationMap[val] = undelegationMap[val].Add(expectedWeights
[val].Quo(w).MulInt(remainingUndelegation).TruncateInt())
            }
            i++
        }
    }

    var msgs []*stakingtypes.MsgUndelegate
    for val, amount := range undelegationMap {
        msgs = append(msgs, &stakingtypes.MsgUndelegate{
            DelegatorAddress: hostChainState.HostAccounts.Delegation.Address,
            ValidatorAddress: val,
            Amount:           sdk.NewCoin(hostChain.BaseDenom, amount),
        })
    }

    return msgs, nil
}
```

However, it impacts the undelegation distribution, the issue is mentioned in another finding.

```
    if !remainingUndelegation.IsZero() {
        w := sdk.ZeroDec()
        for val := range undelegationMap {
            w = w.Add(expectedWeights[val])
        }
        i := 0
        for val := range undelegationMap {
            if i == len(undelegationMap)-1 {
                undelegationMap[val] = undelegationMap[val].Add(
remainingUndelegation)
            } else {
                undelegationMap[val] = undelegationMap[val].Add(expectedWeights
[val].Quo(w).MulInt(remainingUndelegation).TruncateInt())
            }
            i++
        }
    }
```

Recommendation

We advise conducting a thorough review of the logic and rectifying the calculation by subtracting the `totalUndelegation` from the `totalDelegation` variable.

```
totalDelegation = totalDelegation.Sub(totalUndelegation)
```

We also advise applying an absolute value to the weight in order to ensure that the diffAmount always remains positive.

```
diffAmount := weightDiff[i].Weight.Abs().MulInt(totalDelegation).TruncateInt()
```

Alleviation

[Pryzm Team - 09/15/2023] :

The team heeded the advice and resolved this issue in the commit `074a272bb61adb89b9a040ff9dadbc2634e572a8` .

```
totalDelegation = totalDelegation.Sub(totalUndelegation)
```

BRG-01 | INCORRECT UPDATE OF `hostChainState.AmountToBeCompounded`

Category	Severity	Location	Status
Incorrect Calculation	● Medium	bridge_compound.go (374cad8): 81	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_compound.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The `MsgReply()` is invoked during the handle of CompoundBridge's acknowledgment, in which the `hostChainState` will be updated accordingly.

```go
func (b CompoundBridge) MsgReply(ctx sdk.Context, replyData types.ReplyData, _ *sdk.TxMsgData) error {
	var data types.CompoundData
	err := data.Unmarshal(replyData.Data)
	if err != nil {
		return err
	}

	hostChainState, found := b.keeper.GetHostChainState(ctx, replyData.HostChainId)
	if !found {
		return types.ErrHostChainNotFound
	}

	// update AmountToBeDelegated with the accrued rewards and subtract them from AmountToBeCompounded
	hostChainState.AmountToBeDelegated = hostChainState.AmountToBeDelegated.Add(data.CompoundAmount)
	hostChainState.AmountToBeCompounded = hostChainState.AmountToBeCompounded.Sub(data.CompoundAmount)
	hostChainState.HostAccounts.Sweep.Balance = hostChainState.HostAccounts.Sweep.Balance.Add(data.FeeAmount)
	b.keeper.setHostChainState(ctx, hostChainState)

	return b.keeper.SetHostChainIdle(ctx, replyData.HostChainId)
}
```

However, the `hostChainState.AmountToBeCompounded` only subtracts the `CompoundAmount` , which should also subtract the `FeeAmount` because both `CompoundAmount` and `FeeAmount` come from the `AmountToBeCompounded` .

```go

	// add a bank send message to transfer fee amount from rewards to the sweep account
	fee := b.keeper.YieldFeeRatio(ctx, hostChain.GetID()).MulInt(rewardAmount).Ceil().TruncateInt()
	if fee.IsPositive() {
		msgs = append(msgs, &banktypes.MsgSend{
			FromAddress: hostChainState.HostAccounts.Reward.Address,
			ToAddress:   hostChainState.HostAccounts.Sweep.Address,
			Amount:      sdk.NewCoins(sdk.NewCoin(hostChain.BaseDenom, fee)),
		})
	}


	// add a bank send message to transfer (reward - fee) from rewards to the delegation account to be delegated in the next epoch
	compound := rewardAmount.Sub(fee)
```

Recommendation

Recommend subtracting the `hostChainState.AmountToBeCompounded` with both `data.CompoundAmount` and `data.FeeAmount` .

Alleviation

[Pryzm Team - 09/13/2023] :

The team resolved the finding by removing the unused `AmountToBeCompounded` from the host chain state in the commit `dfdad10e973e4a9f954017608763bc70b32ac896` .

CLC-01 MISCONFIGURED TRANSACTION COMMANDS ARE BLOCKED IN ICSTAKING MODULE

Category	Severity	Location	Status
Volatile Code	● Medium	tx_instant_unstake.go (374cad8): 17, 25; tx_redeem_unstaked.go (374cad8): 19, 27; tx_stake.go (374cad8): 17; tx_unstake.go (374cad8): 17	● Resolved

Description

Files:

- `x/icstaking/client/cli/tx_instant_unstake.go`
- `x/icstaking/client/cli/tx_redeem_unstaked.go`
- `x/icstaking/client/cli/tx_stake.go`
- `x/icstaking/client/cli/tx_unstake.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

In the icstaking module, the following 4 commands will be disabled due to the misconfiguration of the arguments and incorrect parse of these arguments.

1. The command `CmdInstantUnstake()` is intended to perform instant unstaking with 4 arguments, while it only requires 3 and the `args[2]` has been used for both `argMinCAmount` and `argMaxCAmount`.

```
13  func CmdInstantUnstake() *cobra.Command {
14      cmd := &cobra.Command{
15          Use:
"instant-unstake [host-chain] [transfer-channel] [min-c-amount] [max-c-amount]",
16          Short: "Broadcast message instant-unstake",
17          Args:  cobra.ExactArgs(3),
18          RunE: func(cmd *cobra.Command, args []string) (err error) {
19              argHostChain := args[0]
20              argTransferChannel := args[1]
21              argMinCAmount, ok := sdk.NewIntFromString(args[2])
22              if !ok {
23                  return sdkerrors.Wrapf(types.ErrInvalidAmount,
"MinCAmount %s cannot be converted to int", args[1])
24              }
25              argMaxCAmount, ok := sdk.NewIntFromString(args[2])
26  ...
```

2. Similarly, the command `CmdRedeemUnstaked()` needs 4 arguments but it only requires 2. Moreover, the `argEpoch` is parsed from the input `u-amount`, which should be the argument `epoch`.

```go
func CmdRedeemUnstaked() *cobra.Command {
    cmd := &cobra.Command{
        Use:
"redeem-unstaked [host-chain] [transferChannel] [u-amount] [epoch]",
        Short: "Broadcast message redeem-unstaked",
        Args:  cobra.ExactArgs(2),
        RunE: func(cmd *cobra.Command, args []string) (err error) {
            argHostChain := args[0]
            argTransferChannel := args[1]
            argUAmount, ok := sdk.NewIntFromString(args[2])
            if !ok {
                return sdkerrors.Wrapf(types.ErrInvalidAmount,
"amount %s cannot be converted to int", args[1])
            }
            argEpoch, err := strconv.ParseUint(args[2], 10, 64)
...
```

3. The command `CmdStake()` requires 3 arguments, but it only accepts 2.

```go
func CmdStake() *cobra.Command {
    cmd := &cobra.Command{
        Use:   "stake [host-chain] [transfer-channel] [amount]",
        Short: "Broadcast message stake",
        Args:  cobra.ExactArgs(2),
...
```

4. Same as above, the command `CmdUnstake()` needs 3 arguments, but it only accepts 2.

```go
func CmdUnstake() *cobra.Command {
    cmd := &cobra.Command{
        Use:   "unstake [host-chain] [transfer-channel] [amount]",
        Short: "Broadcast message unstake",
        Args:  cobra.ExactArgs(2),
...
```

Additionally, the arguments passed in the error messages are incorrect.

Proof of Concept

To reproduce the error, we take the command `instant-unstake` as an example :

1. run the tx command `instant-unstake` with the following inputs:

> prismd tx icstaking instant-unstake 1 2 100 110 --from=prism1vw2jqkgcs4phugu4g29glu3mhvp8yzagzyesdl -y --gas=200000 --fees=10000uprism

2. it returns an error:

> Error: accepts 3 arg(s), received 4

Recommendation

Recommend configuring the correct arguments and parsing the corresponding arguments correctly. In addition, recommend correcting the error messages in the aforementioned commands.

Alleviation

[Pryzm Team - 09/13/2023] :
The team heeded the advice and resolved this issue in the commit `c7a8bf26f5bc8841889f5f28d707a3920bf49997` .

CLI-01 MISCONFIGURATION OF EXPECTED ARGUMENTS BLOCKS THE COMMANDS `CmdIntroduceYammLpToWeightedPool()` AND `CmdSetJoinExitProtocolFee()`

Category	Severity	Location	Status
Volatile Code	● Medium	tx_introduce_yamm_lp_to_weighted_pool.go (374cad8): 18; tx_set_join_exit_protocol_fee.go (374cad8): 20	● Resolved

Description

Files:

- `x/amm/client/cli/tx_introduce_yamm_lp_to_weighted_pool.go`
- `x/amm/client/cli/tx_set_join_exit_protocol_fee.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

Misconfiguration of the expected arguments in the tx commands `CmdIntroduceYammLpToWeightedPool()` and `CmdSetJoinExitProtocolFee()` will disable the execution of these commands.

The tx command `CmdIntroduceYammLpToWeightedPool()` is used to submit a tx to introduce a yamm lp to the the weighted pool, which requires 3 arguments, `weighted-pool-id` , `yamm-pool-id` , `token-normalized-weight` .

However, it expects only 2 arguments in line 18 of the function `CmdIntroduceYammLpToWeightedPool()` :

x/amm/client/cli/tx_introduce_yamm_lp_to_weighted_pool.go

```
14  func CmdIntroduceYammLpToWeightedPool() *cobra.Command {
15      cmd := &cobra.Command{
16          Use:
"introduce-yamm-lp-to-weighted-pool [weighted-pool-id] [yamm-pool-id] [token-
normalized-weight]"
,
17          Short: "Broadcast message introduce-yamm-lp-to-weighted-pool",
18          Args:  cobra.ExactArgs(2),
19          RunE: func(cmd *cobra.Command, args []string) (err error) {
20  ...
```

Similarly, the tx command `CmdSetJoinExitProtocolFee()` is intended to set the join exit protocol fee, which only needs one argument `pool-id` , but it expects 2 arguments as shown in line 20 of the function `CmdSetJoinExitProtocolFee()` :

x/amm/client/cli/tx_set_join_exit_protocol_fee.go

```
func CmdSetJoinExitProtocolFee() *cobra.Command {
    cmd := &cobra.Command{
        Use:   "set-join-exit-protocol-fee [pool-id]",
        Short: "Broadcast message set-join-exit-protocol-fee",
        Args:  cobra.ExactArgs(2),
...
```

As a result, both commands will never be executed successfully.

Proof of Concept

To reproduce the error, please follow the steps:

1. run the tx command `introduce-yamm-lp-to-weighted-pool` with the following inputs:

> prismd tx amm introduce-yamm-lp-to-weighted-pool 1 2 1.0 --from=prism1vw2jqkgcs4phugu4g29glu3mhvp8yzagzyesdl -y --gas=200000 --fees=10000uprism

2. it returns an error:

> Error: accepts 2 arg(s), received 3

Recommendation

Recommend changing the number of expected arguments as follows:

x/amm/client/cli/tx_introduce_yamm_lp_to_weighted_pool.go

```
Args: cobra.ExactArgs(3),
```

x/amm/client/cli/tx_set_join_exit_protocol_fee.go

```
Args:  cobra.ExactArgs(1),
```

Alleviation

[Pryzm Team - 09/15/2023] :

The team removed the two commands to resolve this issue in the commit `96f8395e248f6870647e3a1a52e08acb85d08293`.

CLI-02 | MISSING FEERATIO FLAG IN THE COMMANDS `CmdSetJoinExitProtocolFee()` AND `CmdSetSwapProtocolFee()`

Category	Severity	Location	Status
Volatile Code	● Medium	tx_set_join_exit_protocol_fee.go (374cad8): 58; tx_set_swap_protocol_fee.go (374cad8): 58	● Resolved

Description

Files:

- `x/amm/client/cli/tx_set_join_exit_protocol_fee.go`
- `x/amm/client/cli/tx_set_swap_protocol_fee.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The tx command `CmdSetJoinExitProtocolFee()` is used to set the join exit protocol fee, which requires user's input of `FeeRatio` as the flag. However, this flag has not been added to the command, which disables the execution of the tx command.

x/amm/client/cli/tx_set_join_exit_protocol_fee.go

```
58      flags.AddTxFlagsToCmd(cmd)
```

Similarly, the tx command `CmdSetSwapProtocolFee()` is used to set the swap protocol fee that needs the `FeeRatio` from the flag, but there is no such flag set in the command.

x/amm/client/cli/tx_set_swap_protocol_fee.go

```
58      flags.AddTxFlagsToCmd(cmd)
```

Proof of Concept

To reproduce the error, please follow the steps:

1. submit tx set-join-exit-protocol-fee via the command:

> prismd tx amm set-join-exit-protocol-fee 1 --fee-ratio=0.01 --from=prism1vw2jqkgcs4phugu4g29glu3mhvp8yzagzyesdl -y --gas=200000 --fees=10000uprism

2. it returns an error:

> Error: unknown flag: --fee-ratio

Recommendation

Recommend adding the flag to the aforementioned commands.

Alleviation

[Pryzm Team - 09/15/2023] :

The team removed the two commands to resolve this issue in the commit `96f8395e248f6870647e3a1a52e08acb85d08293` .

EXP-01 | FAILURE OF EXPORTING GENESIS FILE CAUSED BY FETCHING VALIDATOR ADDRESS INCORRECTLY

Category	Severity	Location	Status
Volatile Code	● Medium	export.go (374cad8): 163	● Resolved

Description

Files:

- `app/export.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

According to the current implementation, the function `prepForZeroHeightGenesis()` retrieves an invalid validator address from the KV store, which may be causing the failure to export the genesis file.

The function `prepForZeroHeightGenesis()` is intended to export the genesis state from the Pryzm chain, which includes getting and exporting the validator information from the KV store.

```
156      // Iterate through validators by power descending, reset bond heights, and
157      // update bond intra-tx counters.
158      store := ctx.KVStore(app.keys[stakingtypes.StoreKey])
159      iter := sdk.KVStoreReversePrefixIterator(store, stakingtypes.ValidatorsKey)
160      counter := int16(0)
161
162      for ; iter.Valid(); iter.Next() {
163          addr := sdk.ValAddress(iter.Key()[1:])
164          validator, found := app.StakingKeeper.GetValidator(ctx, addr)
165          if !found {
166              panic("expected validator, not found")
167          }
168
169          validator.UnbondingHeight = 0
170          if applyAllowedAddrs && !allowedAddrsMap[addr.String()] {
171              validator.Jailed = true
172          }
173
174          app.StakingKeeper.SetValidator(ctx, validator)
175          counter++
176      }
177  ...
```

In line 163, the validator address is fetched via `iter.Key()[1:]`, which excludes the `ValidatorsKey`, but it still includes the prefix, the length of the address. This can be demonstrated in the function `GetValidatorKey()` in Cosmos SDK:

x/staking/types/keys.go

```go
// GetValidatorKey creates the key for the validator with address
// VALUE: staking/Validator
func GetValidatorKey(operatorAddr sdk.ValAddress) []byte {
    return append(ValidatorsKey, address.MustLengthPrefix(operatorAddr)...)
}
```

Therefore, the `iter.Key()[1:]` returns the bytes of the length of the address plus the address, and it cannot be correctly converted to the validator address via the function `sdk.ValAddress()`. In this case, it will cause panic because it cannot be found in the staking keeper.

Proof of Concept

To reproduce the panics error, please follow the steps:

1. Start a local node:

> pryzmd start

2. Stop the node;
3. Export the genesis file:

> pryzmd export --for-zero-height > test_genesis.json

Test result:

```
panic: expected validator, not found

goroutine 1 [running]:
github.com/pryzm-finance/pryzm-core/app.(*App).prepForZeroHeightGenesis(_,
{{0x103877518, 0xc0002fa0a0}, {0x103890fd0, 0xc001a05e80}, {{0x0, 0x0}, {0x0, 0x0},
0x22, ...}, ...}, ...)
        github.com/pryzm-finance/pryzm-core/app/export.go:166 +0xf85
github.com/pryzm-finance/pryzm-core/app.
(*App).ExportAppStateAndValidators(0xc000df8a00, 0x1, {0x1050b7090, 0x0, 0x0},
{0x1050b7090, 0x0, 0x0})
        github.com/pryzm-finance/pryzm-core/app/export.go:30 +0x17f
github.com/pryzm-finance/pryzm-
core/cmd/pryzmd/cmd.appCreator.appExport({{{0x1038848e0, 0xc0008cae20},
{0x10389d0c0, 0xc000e199c0}, {0x10388d100, 0xc0005cbcc0}, 0xc0005d8748}},
{0x1038777f0, 0xc001879860}, {0x1038918d0, ...}, ...)
        github.com/pryzm-finance/pryzm-core/cmd/pryzmd/cmd/root.go:293 +0x2ce
github.com/cosmos/cosmos-sdk/server.ExportCmd.func1(0xc0015af800, {0xc0017bf050?,
0x0?, 0x1?})
        github.com/cosmos/cosmos-sdk@v0.47.2/server/export.go:73 +0x435
github.com/spf13/cobra.(*Command).execute(0xc0015af800, {0xc0017bf030, 0x1, 0x1})
        github.com/spf13/cobra@v1.7.0/command.go:940 +0x862
github.com/spf13/cobra.(*Command).ExecuteC(0xc001549500)
        github.com/spf13/cobra@v1.7.0/command.go:1068 +0x3bd
github.com/spf13/cobra.(*Command).Execute(...)
        github.com/spf13/cobra@v1.7.0/command.go:992
github.com/spf13/cobra.(*Command).ExecuteContext(...)
        github.com/spf13/cobra@v1.7.0/command.go:985
github.com/cosmos/cosmos-sdk/server/cmd.Execute(0x1022a0c30?, {0x0, 0x0},
{0xc0012730b0, 0x14})
        github.com/cosmos/cosmos-sdk@v0.47.2/server/cmd/execute.go:32 +0x179
main.main()
        github.com/pryzm-finance/pryzm-core/cmd/pryzmd/main.go:14 +0x30
```

The result outputs the error message: `panic: expected validator, not found`, which means the validator address has not been found.

Recommendation

Recommend using the following function from `CosmosSDK/x/staking/types/keys.go` instead of `iter.Key()[1:]`:

```
// AddressFromValidatorsKey creates the validator operator address from
ValidatorsKey

func AddressFromValidatorsKey(key []byte) []byte {
    kv.AssertKeyAtLeastLength(key, 3)
    return key[2:] // remove prefix bytes and address length
}
```

Alleviation

[Pryzm Team - 09/13/2023] :

The team heeded the advice and resolved the finding by using the recommended function in the commit `3d9393ccaf745d0996bc2ad1adc69815cd4cd8a6` .

MSG-01 | FEE IS COLLECTED FROM USER'S ADDRESS INSTEAD OF REDEEM ACCOUNT

Category	Severity	Location	Status
Logical Issue	● Medium	msg_server_redeem_unstaked.go (374cad8): 61	● Resolved

Description

Files:

- `x/icstaking/keeper/msg_server_redeem_unstaked.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The function `RedeemUnstaked()` is used to redeem user's uAsset to the equivalent amount of underlying asset, during which the unstaking fee will be charged.

```
58      // take the unstaking fee
59      amountCoin := sdk.NewCoin(hostChain.IbcDenom(transferChannel), amount)
60      feeRatio := k.UnstakingFeeRatio(ctx, hostChainId)
61      fee, amountCoin, err := k.treasuryKeeper.CollectFeeByRatio(ctx, address,
 feeRatio, amountCoin, types.UnstakeFeeType)
62      if err != nil {
63          return nil, err
64      }
65
66      // send the remaining amount to user's account
67      err = k.bankKeeper.SendCoinsFromModuleToAccount(ctx, types.
RedeemAccountName, address, sdk.NewCoins(amountCoin))
68      if err != nil {
69          return nil, err
70      }
71  ...
```

The fee is charged via the function `k.treasuryKeeper.CollectFeeByRatio()` :

x/treasury/keeper/keeper_fee_payment.go

```
func (k Keeper) CollectFeeByRatio(ctx sdk.Context, from sdk.AccAddress,
ratio sdk.Dec, amount sdk.Coin, feeType string) (feeCollected sdk.Coin,
remaining sdk.Coin, err error) {
    feeCoin, remCoin, err := k.ComputeFeeByRatio(amount, ratio)
    if err != nil {
        return sdk.Coin{}, sdk.Coin{}, err
    }

    err = k.CollectFee(ctx, from, feeCoin, feeType)
    if err != nil {
        return feeCollected, remaining, err
    }

    return feeCoin, remCoin, nil
}
```

However, the fee is incorrectly deducted from the user's address, which should be from the redeem account. Since the user's address could possibly do not have such fee, then this operation fails.

Recommendation

Recommend charging the unstaking fee from the redeem account.

Alleviation

[Pryzm Team - 09/13/2023] :

The team heeded the advice and resolved the finding by collecting the redeem fee from the redeem account in the commit `beed73d18d1c0d8b76013ed88d62260597a59925` .

```
    redeemAccountAddress :=
k.accountKeeper.GetModuleAddress(types.RedeemAccountName)
    fee, amountCoin, err := k.treasuryKeeper.CollectFeeByRatio(ctx,
redeemAccountAddress, feeRatio, amountCoin, types.UnstakeFeeType)
```

ORA-02 VARIABLE `hostChainState.AmountToBeCompounded` USED TO COMPUTE THE EXCHANGE RATE INCLUDES PROTOCOL FEE

Category	Severity	Location	Status
Logical Issue	● Medium	oracle_callback.go (374cad8): 92	● Resolved

Description

Files:

- `x/icstaking/keeper/oracle_callback.go`
- `x/icstaking/keeper/bridge_compound.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The exchange rate is updated in the function `OnMajorityVote()` according to the following formula:

$$\text{hostChainState.ExchangeRate} = \frac{\text{total delegation} + \text{delegation queue} + \text{hostChainState.AmountToBeCompounded} + \text{hostChainState.AmountToBeDelegated}}{\text{totalCTokenSupply}}$$

x/icstaking/keeper/oracle_callback.go

```
  86      totalCTokenSupply := c.k.bankKeeper.GetSupply(ctx, hostChain.CDenom()).
Amount
  87      if !totalCTokenSupply.IsZero() {
  88          oldER := hostChainState.ExchangeRate
  89          hostChainState.ExchangeRate = sdk.NewDecFromInt(
  90              totalDelegation.
  91                  Add(delegationQueueAmount).
  92                  Add(hostChainState.AmountToBeCompounded).
  93                  Add(hostChainState.AmountToBeDelegated)).
  94              QuoInt(totalCTokenSupply)
  95          err := c.k.exchangeRateListeners.ExchangeRateUpdated(ctx, hostChainId,
&oldER, hostChainState.ExchangeRate)
  96          if err != nil {
  97              return err
  98          }
  99      }
```

in which the term `hostChainState.AmountToBeCompounded` includes both the protocol fee and the amount to be compounded. It will be deducted in the function `c.k.compoundBridge.compoundRewards()` :

x/icstaking/keeper/oracle_callback.go

```go
	if !payload.RewardAccountBalance.IsZero() {
		return c.k.compoundBridge.compoundRewards(ctx, hostChain,
hostChainState.AmountToBeCompounded)
	}
```

That means the amount will be deducted as the protocol fee contributes to the delegation.

x/icstaking/keeper/bridge_compound.go

```go
func (b CompoundBridge) compoundRewards(ctx sdk.Context, hostChain types.
HostChain, rewardAmount math.Int) error {
	hostChainState, found := b.keeper.GetHostChainState(ctx, hostChain.GetID())
	if !found {
		return types.ErrHostChainNotFound
	}

	hostChainState.State = types.State_COMPOUNDING
	b.keeper.setHostChainState(ctx, hostChainState)

	var msgs []sdk.Msg


// add a bank send message to transfer fee amount from rewards to the sweep account
	fee := b.keeper.YieldFeeRatio(ctx, hostChain.GetID()).MulInt(rewardAmount).
Ceil().TruncateInt()
	if fee.IsPositive() {
		msgs = append(msgs, &banktypes.MsgSend{
			FromAddress: hostChainState.HostAccounts.Reward.Address,
			ToAddress:   hostChainState.HostAccounts.Sweep.Address,
			Amount:      sdk.NewCoins(sdk.NewCoin(hostChain.BaseDenom, fee)),
		})
	}


// add a bank send message to transfer (reward - fee) from rewards to the delegation
account to be delegated in the next epoch

	compound := rewardAmount.Sub(fee)
	if compound.IsPositive() {
		msgs = append(msgs, &banktypes.MsgSend{
			FromAddress: hostChainState.HostAccounts.Reward.Address,
			ToAddress:   hostChainState.HostAccounts.Delegation.Address,
			Amount:      sdk.NewCoins(sdk.NewCoin(hostChain.BaseDenom, compound
)),
		})
	}
```

Recommendation

The auditing team would like to check with the PRYZM team if the calculation of the exchange rate should exclude the protocol fee.

Alleviation

[Pryzm Team - 09/13/2023] :

The team resolved the finding by excluding the protocol fee from the exchange rate calculation in the commit `f67a16421f3b1479deff1d57189dbfd1f33bebe1` .

ABC-01 | HEAVY COMPUTATION IN ICSTAKING'S BEGINBLOCKER COULD SLOW DOWN BLOCK PRODUCTION

Category	Severity	Location	Status
Volatile Code, Denial of Service	● Minor	x/icstaking/keeper/abci.go (pryzm-core-0c34472): 10~13	● Resolved

Description

Files:

- `x/icstaking/keeper/abci.go`

Commit:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The function `BeginBlocker()` is executed at the beginning of each block, which is supposed to have light or constant computation load in order to not impact the block production. If the `BeginBlocker()` contains too heavy computation, it will lead to slow block production or even exceed the block propose timeout. In this case, it is possible to halt the chain.

The icstaking `BeginBlocker()` iterate all the host chains to execute the batched delegation and undelegation. Though the registration of host chain is performed via governance, the loop is unbonded and could lead to unexpected block production slow down.

```
10  func (k Keeper) BeginBlocker(ctx sdk.Context) {
11      // iterate all host chains and manage delegation and undelegation
12      hostChains := k.GetAllHostChain(ctx)
13      for _, hostChain := range hostChains {
14  ...
```

Recommendation

Recommend limiting the number of host chains registered on Pryzm.

Alleviation

[Pryzm Team - 10/25/2023] :

The team resolved the finding by limiting the number of host chains to a maximum of 100 to resolve this issue.

The change is reflected in the commit `31518a6e6342a9b1880feedf94d53bda2bf11581`.

ASS-01 | VALIDATION OF GENESIS STATE IN `assets` MODULE

Category	Severity	Location	Status
Volatile Code	● Minor	module.go (374cad8): 126~134; genesis.go (374cad8): 35~48	● Resolved

Description

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

Lack of Validation When Initializing Genesis State

Files:

- `x/assets/module.go`
- `x/assets/types/genesis.go`
- `x/assets/keeper/genesis.go`

The method `GenesisState.Validate()` is used to validate the genesis state for the module; it will be called by the method `AppModuleBasic.ValidateGenesis()` when using the commands `validate-genesis` and `gentx` .

However, this method is not called when initializing the module by calling the method `AppModule.InitGenesis()` or the function `InitGenesis()` in the file `x/assets/keeper/genesis.go` , which will import an incorrect genesis state and may cause a potential failure.
e.g., the RefractableAsset whose `id` is **"c:token"** can be set in the genesis.json file and imported into the genesis state into the module(A RefractableAsset's id can not contain a ":").

Lack of Validation for `ExchangeRate`

Files:

- `x/assets/types/genesis.go`

The list of `ExchangeRate` can be passed via the genesis.json file. In the `GenesisState.Validate()` method, there is a validation for ensuring the `AssetId` of all elements is not duplicated. But the value of `ExchangeRate.Rate` is not validated, which means the command `validate-genesis` can't find the error even if a rate is invalid.

Validation for `RefractableAsset.TokenDenom`

File: `x/assets/types/refractable_asset.go`

For a RefractableAsset, if the `HostChainId` of it is not empty, the `TokenDenom` of it must be empty. Otherwise, the method `RefractableAsset.Validate()` will return an error.

```go
        icstaked := strings.TrimSpace(a.HostChainId) != ""
        if icstaked && a.TokenDenom != "" {
                return sdkerrors.Wrapf(ErrInvalidTokenDenom,
"token denom must be empty if the asset is icstaked by PRISM")
        } else if !icstaked {
                if err := sdk.ValidateDenom(a.TokenDenom); err != nil {
                        return sdkerrors.Wrapf(ErrInvalidTokenDenom, err.Error(
))
                }
        }
```

If this method is called when registering new assets by the message `RegisterAsset`, this validation will make sense because the value of `TokenDenom` field will be set by the `CDenom` which is queried from the host chain: `x/assets/keeper/msg_server_register_asset.go`

```go
        if asset.IsICStaked() {
                hostChain, found := k.icstakingKeeper.GetHostChain(ctx, asset.
HostChainId)
                if !found {
                        return nil, sdkerrors.Wrapf(icstakingtypes.
ErrHostChainNotFound, "host chain not found for icstaked asset %s", asset.GetID())
                }
                asset.TokenDenom = hostChain.CDenom()
        } else {
                // check that the token denom is valid
                err := sdk.ValidateDenom(asset.TokenDenom)
                if err != nil {
                        return nil, sdkerrors.Wrapf(types.ErrInvalidTokenDenom,
 "token denom is not valid: %s", err.Error())
                }
        }
```

But if this validation is used in the module genesis, it will cause an unreasonable error. If the module allows users to add external tokens since the module doesn't allow us to update the registered tokens, we must set both the `TokenDenom` and `HostChainId` fields in the genesis state, and this will cause the error **"token denom must be empty if the asset is icstaked by PRYZM"** when we validate the genesis state by commands `validate-genesis` and `gentx`.

Recommendation

We recommend adding the validation for the genesis state when initializing the genesis state and the validation for `ExchangeRate` when validating the genesis state to ensure the value of `ExchangeRate` is in a valid range.

Also, we recommend the client consider whether an external token can be registered when initializing the genesis state. If the protocol allows registering, since both the `TokenDenom` and `HostChainId` fields in the genesis state must be set, a genesis state which contains an external token can't pass the validation by the commands `validate-genesis` and `gentx`.

Alleviation

[Pryzm Team - 09/18/2023]: The team heeded the advice and resolved this issue in the commit 0c34472f03010ddc4048ba0727c33c6418d69c2a.

BRE-01 | DISCUSSION ON THE DELEGATION REBALANCE LOGIC

Category	Severity	Location	Status
Logical Issue	● Minor	bridge_redelegate.go (374cad8): 161~168	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_redelegate.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The function `createRedelegationMsgs()` is used to create the redelegation messages, which contain the rebalance of delegation that works as follows:

1. calculate the total delegation from PRYZM to the validators on the host chain;
2. compute the actual weight via the validator's delegated amount / total delegation;
3. take the weight difference with their respective expected weights;
4. sort the weight difference in ascending order;
5. use two pointers from the lowest and highest indices of the slice, weight difference;
6. take the absolute value of the weight difference with a low index (as it's negative);
7. if the difference between the weight difference of low and high indices is larger than the rebalance threshold, move to the next one.

```go
    totalDelegation := sdk.ZeroInt()
    expectedWeights := make(map[string]sdk.Dec, len(hostChain.Validators))
    for _, v := range hostChain.Validators {
        expectedWeights[v.Address] = v.Weight
        valInfo, found := hostChainState.Validators[v.Address]
        if !found {
            continue
        }
        totalDelegation = totalDelegation.Add(valInfo.DelegatedAmount)
    }

    weightDiff := make([]types.Validator, len(hostChain.Validators))
    for i, v := range hostChain.Validators {
        valInfo, found := hostChainState.Validators[v.Address]
        if !found {
            continue
        }
        actualWeight := sdk.NewDecFromInt(valInfo.DelegatedAmount).QuoInt(
totalDelegation)
        weightDiff[i] = types.Validator{
            Address: v.Address,
            Weight:  actualWeight.Sub(expectedWeights[v.Address]),
        }
    }

    // ascending sort
    sort.SliceStable(weightDiff, func(i, j int) bool {
        return weightDiff[i].Weight.LT(weightDiff[j].Weight)
    })

    rebalanceThreshold := k.RebalanceThreshold(ctx, hostChain.GetID())
    minRebalanceAmount := k.MinRebalanceAmount(ctx, hostChain.GetID())
    maxRedelegateMsgs := k.MaxRedelegationMsgs(ctx, hostChain.GetID())

    // FIXME change var names
    var redelegateMsgs []sdk.Msg
    underIndex := 0
    overIndex := len(weightDiff) - 1

    for i := int32(0); i < maxRedelegateMsgs; i++ {
        if underIndex == overIndex {
            break
        }

        underWeight := weightDiff[underIndex].Weight.Abs()
        underValidator := weightDiff[underIndex].Address

        overWeight := weightDiff[overIndex].Weight
        overValidator := weightDiff[overIndex].Address

        var diff sdk.Dec
        if underWeight.LT(overWeight) {
            diff = overWeight.Sub(underWeight)
            underIndex++
        } else {
            diff = underWeight.Sub(overWeight)
            overIndex--
        }


// check that the re-delegation amount is more than the rebalance threshold
        if diff.LT(rebalanceThreshold) {
            break
        }
        redelegationAmount := diff.MulInt(totalDelegation).TruncateInt()
        if redelegationAmount.LT(minRebalanceAmount) {
            break
        }

        redelegateMsgs = append(redelegateMsgs, &stakingtypes.
MsgBeginRedelegate{
            DelegatorAddress:    hostChainState.HostAccounts.Delegation.Address
,
            ValidatorSrcAddress: overValidator,
            ValidatorDstAddress: underValidator,
            Amount:              sdk.NewCoin(hostChain.BaseDenom,
 redelegationAmount),
        })
    }
```

However, the condition that the difference between weight differences should be larger than the rebalance threshold does not align with the documentation:

> This message allows any permissionless users to request rebalancing of delegations for host chain validators. This message executes the process of rebalancing only if a specific time has passed from the last rebalancing and the divergence of delegations from validators expected weights is more than a specific threshold.

Recommendation

The auditing team thinks this condition should be both weight differences are larger than the rebalance threshold to perform the redelegation, not the difference being larger than the rebalance threshold.

Alleviation

[Pryzm Team - 09/13/2023] :

The team resolved the finding by refactoring the delegation rebalance logic to align with the design in the commit `21b11ae1fab27b1befa9d6017de2ed6ffc499a31` .

[CertiK - 09/13/2023] :

The refactored logic in the commit could lead to the following scenario that the updated weight diff is less than the `rebalanceThreshold` . As a result, it exits the for loop earlier.

1. For example, suppose the weight diffs in the ascending order are: -0.21, -0.11, ..., 0.11, 0.2, and the `rebalanceThreshold` is 0.1.
2. After the first update of weight diff, the the overweight index is decremented, and the underweight index is still 0 with updated weight diff -0.21 + 0.2 = -0.01, of which the absolute value is less than `rebalanceThreshold`.
3. In this case, it exits earlier from the for loop though the second one with `-0.11` is legit for rebalance.

```go
    // create a mapping of validators to the diff of their current weight to their
expected weight
    weightDiff := make([]types.Validator, len(expectedWeights))
    i := 0
    for valAddress, expectedWeight := range expectedWeights {
        actualWeight := sdk.ZeroDec()
        valInfo, found := hostChainState.Validators[valAddress]
        if found {
            actualWeight =
sdk.NewDecFromInt(valInfo.DelegatedAmount).QuoInt(totalDelegation)
        }
        weightDiff[i] = types.Validator{
            Address: valAddress,
            Weight:  actualWeight.Sub(expectedWeight),
        }
        i++
    }

    // ascending sort, so validators with less delegation than expected are first
    sort.SliceStable(weightDiff, func(i, j int) bool {
        return weightDiff[i].Weight.LT(weightDiff[j].Weight)
    })

    underWeightIndex := 0
    overWeightIndex := len(weightDiff) - 1

    for i := int32(0); i < maxRedelegateMsgs; i++ {
        if underWeightIndex == overWeightIndex {
            break
        }

        underWeightDiff := weightDiff[underWeightIndex].Weight.Abs()
        underWeightValidator := weightDiff[underWeightIndex].Address

        overWeightDiff := weightDiff[overWeightIndex].Weight
        overWeightValidator := weightDiff[overWeightIndex].Address

        // calculate the weight that can be re-delegated to make one side reach
balance
        var dw sdk.Dec
        if underWeightDiff.LT(overWeightDiff) {
            dw = underWeightDiff
            underWeightIndex++
            // subtract dw from the over weight index for the next iteration
            weightDiff[overWeightIndex] = types.Validator{
                Address: overWeightValidator,
                Weight:  overWeightDiff.Sub(dw),
            }
        } else {
            dw = overWeightDiff
            overWeightIndex--
            // subtract dw from the under weight index for the next iteration
            weightDiff[underWeightIndex] = types.Validator{
                Address: underWeightValidator,
                Weight:  underWeightDiff.Sub(dw).Neg(),
            }
        }

        // check that the re-delegation amount is more than the rebalance threshold
        if dw.LT(rebalanceThreshold) {
            break
        }

        redelegationAmount := dw.MulInt(totalDelegation).TruncateInt()
        if redelegationAmount.LT(minRebalanceAmount) {
            break
        }
```

[Pryzm Team - 09/22/2023] :

The team resolved the issue of early exit from the for loop by proceeding to the next one if the updated weight is less than rebalanceThreshold in the commit `ee6472deee9fe6b7e653fef747de7c4c7a94feb6` .

BRG-02 | NON-GUARANTEED HOST CHAIN STATE

Category	Severity	Location	Status
Logical Issue	● Minor	bridge_compound.go (374cad8): 25	● Resolved

Description

Files:

- `x/icstaking/keeper/bridge_compound.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

There is no guarantee that the state of the host chain is idle before the function `compoundRewards` is invoked.

Recommendation

We recommend adding the validation to ensure the host chain is idle before calling the function `compoundRewards`.

```
    if hostChainState.State != types.State_IDLE {
        return nil, sdkerrors.Wrap(types.ErrRebalanceNotNeeded, "host chain state is
not IDLE")
    }
```

Alleviation

[Pryzm Team - 09/15/2023] :
The team heeded the advice and applied the host chain state validation to the functions listed below to resolve this issue in the commit `00059106c257e880205cfc6c603e928def058247`.

- x/icstaking/keeper/bridge_compound.go
- x/icstaking/keeper/bridge_delegate.go
- x/icstaking/keeper/bridge_sweep.go
- x/icstaking/keeper/bridge_undelegate.go

CRE-01 | POSSIBLE OVERWRITE OF DENOM METADATA IN GENESIS

Category	Severity	Location	Status
Volatile Code	● Minor	createdenom.go (374cad8): 30	● Resolved

Description

Files:

- `x/tokenfactory/keeper/createdenom.go`
- `app/app.go`
- `x/tokenfactory/keeper/genesis.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The current implementation does not check if the denom metadata has been set in the bank module, which could lead to the denom metadata overwrite in the genesis.

The init of module genesis follows the order defined in the `app.go` , in which the `tokenfactory` is performed after bank module:

app/app.go

```
    app.mm.SetOrderInitGenesis(
        capabilitytypes.ModuleName
        authtypes.ModuleName,
        banktypes.ModuleName,
...
        tokenfactorytypes.ModuleName,
```

In the init of tokenfactory genesis, it invokes the function `createDenomAfterValidation()` to set the denom metadata:

x/tokenfactory/keeper/genesis.go

```go
func (k Keeper) InitGenesis(ctx sdk.Context, genState types.GenesisState) {
    k.CreateModuleAccount(ctx)

    if genState.Params.DenomCreationFee == nil {
        genState.Params.DenomCreationFee = sdk.NewCoins()
    }
    k.SetParams(ctx, genState.Params)

    for _, genDenom := range genState.GetFactoryDenoms() {
        creator, _, err := types.DeconstructDenom(genDenom.GetDenom())
        if err != nil {
            panic(err)
        }
        err = k.createDenomAfterValidation(ctx, creator, genDenom.GetDenom())
        if err != nil {
            panic(err)
        }
        err = k.setAuthorityMetadata(ctx, genDenom.GetDenom(), genDenom.
GetAuthorityMetadata())
        if err != nil {
            panic(err)
        }
    }
}
```

However, the function `createDenomAfterValidation()` does not check if the denom metadata has already been set in the bank module:

x/tokenfactory/keeper/createdenom.go

```go
func (k Keeper) createDenomAfterValidation(ctx sdk.Context, creatorAddr string,
denom string) (err error) {
    denomMetaData := banktypes.Metadata{
        DenomUnits: []*banktypes.DenomUnit{{
            Denom:    denom,
            Exponent: 0,
        }},
        Base: denom,
    }

    k.bankKeeper.SetDenomMetaData(ctx, denomMetaData)
...
```

In case that the same denom has been used as the key in the bank, it will overwrite it.

Reference:

- https://github.com/osmosis-labs/osmosis/pull/5532

Recommendation

Recommend adding an extra check to ensure the denom metadata has not been set in the bank.

Alleviation

[Pryzm Team - 09/13/2023] :

The team heeded the advice and resolved the finding with the fix from Osmosis in the commit `2768e3012759eeacc53896d293ac29614acc8a02` .

CRE-02 | MISSING DISPLAY DENOM WILL FAIL DENOM METADATA VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	createdenom.go (374cad8): 31~37	● Resolved

Description

Files:

- `x/tokenfactory/keeper/createdenom.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The `denomMetaData` created in the function `createDenomAfterValidation()` does not match the metadata defined in the bank module, which will lead to the failure of validation in the genesis state.

The function `createDenomAfterValidation()` creates a denom metadata with the following format:

x/tokenfactory/keeper/createdenom.go

```
	denomMetaData := banktypes.Metadata{
		DenomUnits: []*banktypes.DenomUnit{{
			Denom:    denom,
			Exponent: 0,
		}},
		Base: denom,
	}
```

in which the `Display` is not defined and the DenomUnit only contains the base denom, which does not align with the Metadata struct. The denom metadata will be set in the bank keeper and validated against the following function `Validate()` from the Cosmos SDK in the genesis state.

x/bank/types/metadata.go

```
func (m Metadata) Validate() error {
    if strings.TrimSpace(m.Name) == "" {
        return errors.New("name field cannot be blank")
    }

    if strings.TrimSpace(m.Symbol) == "" {
        return errors.New("symbol field cannot be blank")
    }

    if err := sdk.ValidateDenom(m.Base); err != nil {
        return fmt.Errorf("invalid metadata base denom: %w", err)
    }

    if err := sdk.ValidateDenom(m.Display); err != nil {
        return fmt.Errorf("invalid metadata display denom: %w", err)
    }

    var (
        hasDisplay      bool
        currentExponent uint32 // check that the exponents are increasing
    )

    seenUnits := make(map[string]bool)

    for i, denomUnit := range m.DenomUnits {
        // The first denomination unit MUST be the base
        if i == 0 {
            // validate denomination and exponent
            if denomUnit.Denom != m.Base {
                return fmt.Errorf("metadata's first denomination unit must be the
one with base denom '%s'", m.Base)
            }
            if denomUnit.Exponent != 0 {
                return fmt.Errorf("the exponent for base denomination unit %s must
be 0", m.Base)
            }
        } else if currentExponent >= denomUnit.Exponent {
            return errors.New("denom units should be sorted asc by exponent")
        }

        currentExponent = denomUnit.Exponent

        if seenUnits[denomUnit.Denom] {
            return fmt.Errorf("duplicate denomination unit %s", denomUnit.Denom)
        }

        if denomUnit.Denom == m.Display {
            hasDisplay = true
        }
        if err := denomUnit.Validate(); err != nil {
            return err
        }

        seenUnits[denomUnit.Denom] = true
    }

    if !hasDisplay {
        return fmt.Errorf("metadata must contain a denomination unit with display
denom '%s'", m.Display)
    }

    return nil
}
```

As a result, missing `Display` in the Metadata and DenomUnit will cause the the failure of genesis state validation.

Recommendation

Recommend properly setting the `Display` in Metadata and DenomUnit.

Alleviation

[Pryzm Team - 09/13/2023] :

The team resolved the finding by setting the display denom same as the base denom in the commit `2768e3012759eeacc53896d293ac29614acc8a02` .

FLO-03 THE CLAIMABLE PURCHASED TOKEN AMOUNT DOES NOT CONSIDER `PendingPurchase`

Category	Severity	Location	Status
Logical Issue	● Minor	x/flowtrade/keeper/claim.go (flowtrade): 147; x/flowtrade/types/position.go (flowtrade): 26	● Acknowledged

Description

Files:

- `x/flowtrade/keeper/claim.go`
- `x/flowtrade/types/position.go`

Commit:

- `930876154d4296a366ba2ca179c227c6663cc55b`

The owner of a position has the ability to utilize the `ClaimTokenOut` function in order to retrieve the currently purchased tokens associated with that position. However, it's important to note that the calculation for the claimable token amount does not take into account the `PendingPurchase` value, this includes the amount of purchased tokens that have been paid for but are not included in the calculation due to rounding errors. As a result, the owner of the position may not receive the complete purchased tokens and could potentially experience a loss.

- `x/flowtrade/keeper/claim.go`

```
func (k Keeper) ClaimTokenOut(ctx sdk.Context, flowId uint64, address sdk.AccAddress) (claimed sdk.Coin, fee sdk.Coin, err error) {
    ...

// subtract the already claimed tokens from the purchased token out and calculate the claimable amount and check that's not zero

    claimable := position.PurchasedTokenOut.Sub(position.ClaimedAmount)
    if claimable.IsZero() {
        return claimed, fee, sdkerrors.Wrap(types.ErrZeroClaimable, "flow has no claimable amount")
    }
    ...
}
```

- `x/flowtrade/types/position.go`

```go
func (p *Position) UpdateDistIndex(flow Flow) (updated bool) {
    indexDiff := flow.DistIndex.Sub(p.DistIndex)

    // return if diff is negative or zero
    if !indexDiff.IsPositive() {
        return
    }

    p.DistIndex = flow.DistIndex
    updated = true


// return without updating other fields if no shares are available (flow is empty)
    if flow.TotalShares.IsZero() {
        return
    }

    purchased := indexDiff.MulInt(p.Shares).Add(p.PendingPurchase)
    purchasedTruncated := purchased.TruncateInt()
    purchasedRemainder := purchased.Sub(sdk.NewDecFromInt(purchasedTruncated))

    p.PurchasedTokenOut = p.PurchasedTokenOut.Add(purchasedTruncated)
    p.PendingPurchase = purchasedRemainder

    newInBalance := flow.TokenInBalance.Mul(p.Shares).Quo(flow.TotalShares)
    p.SpentTokenIn = p.SpentTokenIn.Add(p.TokenInBalance.Sub(newInBalance))
    p.TokenInBalance = newInBalance
    return
}
```

Recommendation

It is recommended to take into account the `PendingPurchase` when calculating the claimable purchased tokens and ensure that users are not at risk of suffering losses.

Alleviation

[Pryzm Team - 08/18/2023] :

In `position.UpdateDistIndex` method, the `PendingPurchase` field of the position is set to the truncated decimals of the actual purchased amount and is added to the purchased amount in the next dist index update. The value of this field always remains lower than 1. When a user is claiming their purchased tokens, this decimal value cannot be transferred to the user's account as it's a <1 decimal.

[CertiK - 11/18/2023] :

Upon the review, the severity level of this issue has been reassessed and the severity has been downgraded from Medium to Minor.

HOO-01 | MINT PRYZM EACH EPOCH

Category	Severity	Location	Status
Logical Issue	● Minor	hooks.go (374cad8): 27	● Acknowledged

Description

Files:

- `x/mint/keeper/hooks.go`
- `x/mint/types/minter.go`

Commit:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The Mint module is responsible for minting PRYZM tokens and distributing them to different destinations. A portion of the tokens will be allocated to the FeeCollector and will be calculated and distributed to the stakers (the bonded validators in the staking module) by the distribution module at the beginning of the next block. The Inflation calculation formula is almost the same as the cosmos/sdk/mint module, with the only difference being that in Cosmos, the rewards are minted in every block, while in Pryzm, they are minted at the end of each epoch. This difference may cause the original incentive mechanism to be ineffective.

- `x/mint/keeper/hooks.go`

```
 27  func (k Keeper) AfterEpochEnd(ctx sdk.Context, epochIdentifier string,
epochNumber int64) error {
 28      params := k.GetParams(ctx)
 29      if epochIdentifier != params.EpochIdentifier {
 30          return nil
 31      }
 32      // not distribute rewards if it's not time yet for rewards distribution
 33      if epochNumber < params.MintingRewardsDistributionStartEpoch {
 34          return nil
 35      }
 36
 37      epochsPerYear := int64(yearSeconds / k.epochsKeeper.GetEpochInfo(ctx,
epochIdentifier).Duration.Seconds())
 38      .......
 39  }
```

- `x/mint/types/minter.go`

```
func (m Minter) NextInflationRate(params Params, bondedRatio sdk.Dec,
 epochsPerYear int64) sdk.Dec {
// The target annual inflation rate is recalculated for each previsions cycle. The
// inflation is also subject to a rate change (positive or negative) depending on
// the distance from the desired ratio (67%). The maximum rate change possible is
// defined to be 13% per year, however the annual inflation is capped as between
	// 7% and 20%.

	// (1 - bondedRatio/GoalBonded) * InflationRateChange
	inflationRateChangePerYear := sdk.OneDec().
		Sub(bondedRatio.Quo(params.GoalBonded)).
		Mul(params.InflationRateChange)

	inflationRateChange := inflationRateChangePerYear.Quo(sdk.NewDec(
epochsPerYear)) // This is different from the sdk. It uses the epochs.
	......
}
```

Due to inflation, the rewards minted will be fully distributed in the next block after the end of the epoch. In non-epoch start or end blocks, to the stakers, there are almost no inflation rewards, only some transaction fees. This is unfair and may encourage validators to participate in the committee before the end of the epoch and then exit the committee after the start of the next epoch. This is not conducive to block generation and may result in the ineffectiveness of the incentive mechanism.

Recommendation

We recommend the team reconsider the design.

Alleviation

[Pryzm Team - 10/25/2023] :

We recognize the concern, but in practice, we intend to use brief time periods such as an hour or even half an hour.

HOS-02 | RETURN VALUE OF `GetChannel()` IS NOT HANDLED

Category	Severity	Location	Status
Volatile Code	● Minor	x/icstaking/types/host_chain.go (pryzm-core-0c34472): 29	● Resolved

Description

Files:

- `x/icstaking/types/host_chain.go`

Commit:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The function `GetChannel()` checks if a channel id exists on the host chain.

```
18  func (hostChain HostChain) GetChannel(channelId string) (channel TransferChannel
, found bool) {
19      for _, c := range hostChain.TransferChannels {
20          if c.Id == channelId {
21              return c, true
22          }
23      }
24      return
25  }
```

which is called by the function `IbcDenom()` to create the ibc denom based on the channel id:

```
28  func (hostChain HostChain) IbcDenom(channelId string) string {
29      channel, _ := hostChain.GetChannel(channelId)
30      denom := hostChain.BaseDenom
31      if strings.TrimSpace(channel.WrappedDenom) != "" {
32          denom = channel.WrappedDenom
33      }
34
35      return transfertypes.DenomTrace{
36          Path:      fmt.Sprintf("%s/%s", transfertypes.PortID, channelId),
37          BaseDenom: denom,
38      }.IBCDenom()
39  }
```

However, in line 29, it does not check return value of `hostChain.GetChannel(channelId)` to ensure the channel id exists.

Recommendation

Recommend handling the return value of `hostChain.GetChannel(channelId)` to ensure the ibc denom is created if it does exist.

Alleviation

[Pryzm Team - 10/16/2023]:
The IbcDenom method on HostChain is a utility method for calculating the ibc denom considering the channel name and is not meant to check the existence of the channel; it only does the calculation and returns the ibc denom. The GetChannel method is only called for getting wrapped channels in case of existence. We enhanced the existence checking in the commit `1296f44f3efc6aa605d29eab1ed19dfeb1813850` .

KEE-03 | LACK OF VALIDATION FOR `transferChannel`

Category	Severity	Location	Status
Logical Issue	● Minor	msg_server_instant_unstake.go (374cad8): 45; msg_server_redeem_unstaked.go (374cad8): 33; msg_server_stake.go (374cad8): 36	● Resolved

Description

Files:

- `x/icstaking/keeper/msg_server_redeem_unstaked.go`
- `x/icstaking/keeper/msg_server_stake.go`
- `x/icstaking/keeper/msg_server_instant_unstake.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

There is no guarantee that the transfer channel is the supported transfer channel for transferring the base_denom tokens between the host chain and Pryzm.

And the validation of the transfer channel is missing in the `ValidateBasic()` of messages.

Recommendation

We recommend adding the validation to ensure the transfer channel is the supported transfer channel, adding the validation of the transfer channel in the ValidateBasic() of messages.

Alleviation

[Pryzm Team - 09/15/2023] :
The team heeded the advice and resolved this issue in the commit `aa9d594b97347afb07de9e5d283c8581683cc3af` .

KEE-07 | POTENTIAL DIVISION BY ZERO

Category	Severity	Location	Status
Volatile Code	● Minor	bridge_redelegate.go (374cad8): 129; bridge_undelegate.go (374cad8): 193; msg_server_redeem_unstaked.go (374cad8): 46; oracle_callback.go (374cad8): 120	● Resolved

Description

Files:

- `x/icstaking/keeper/oracle_callback.go`
- `x/icstaking/keeper/msg_server_redeem_unstaked.go`
- `x/icstaking/keeper/bridge_redelegate.go`
- `x/icstaking/keeper/bridge_undelegate.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The linked code does not check the denominator is nonzero when performing the division, which could lead to division by zero panics. Note that there is a check in the function `CreateDelegationMsgs()` :

x/icstaking/keeper/bridge_delegate.go

```
    if totalDeposit.IsZero() {
        return []*stakingtypes.MsgDelegate{}
    }
```

In this case, the division by zero will not occur.

Proof of Concept

To reproduce the error, we take the redelegate as an example:

1. submit a rebalance-delegations transaction:

> pryzmd tx icstaking rebalance-delegations inj --from=pryzm1vw2jqkgcs4phugu4g29glu3mhvp8yzagzyesdl -y --gas=200000 --fees=10000upryzm

2. query the transaction, 78A3FB92CFC06ED78CAC9B423048D8FC09A0D4365B7F1AF4D1CCD9AFBAFDFC66:

```
pryzmd q tx 78A3FB92CFC06ED78CAC9B423048D8FC09A0D4365B7F1AF4D1CCD9AFBAFDFC66
```

Result:

```
raw_log: "recovered: division by zero\nstack:\ngoroutine 89
[running]:\nruntime/debug.Stack()\n\truntime/debug/stack.go:24
  +0x65\ngithub.com/cosmos/cosmos-
sdk/baseapp.newDefaultRecoveryMiddleware.func1({0x1023e0c20,
  0x103843b90})\n\tgithub.com/cosmos/cosmos-sdk@v0.47.2/baseapp/recovery.go:71
+0x27\ngithub.com/cosmos/cosmos-
sdk/baseapp.newRecoveryMiddleware.func1({0x1023e0c20?,
  0x103843b90?})\n\tgithub.com/cosmos/cosmos-sdk@v0.47.2/baseapp/recovery.go:39
+0x30\ngithub.com/cosmos/cosmos-sdk/baseapp.processRecovery({0x1023e0c20,
  0x103843b90}, 0xc00f4bc000?)\n\tgithub.com/cosmos/cosmos-
sdk@v0.47.2/baseapp/recovery.go:28
  +0x37\ngithub.com/cosmos/cosmos-sdk/baseapp.processRecovery({0x1023e0c20,
0x103843b90},
  0x103890fd0?)\n\tgithub.com/cosmos/cosmos-sdk@v0.47.2/baseapp/recovery.go:33
+0x5e\ngithub.com/cosmos/cosmos-sdk/baseapp.
(*BaseApp).runTx.func1()\n\tgithub.com/cosmos/cosmos-
sdk@v0.47.2/baseapp/baseapp.go:632
  +0xf0\npanic({0x1023e0c20, 0x103843b90})\n\truntime/panic.go:890
+0x262\nmath/big.nat.div({0x0?,
  0x100017e05?, 0x0?}, {0x0?, 0x135bf9e18?, 0x10?}, {0x0?, 0x100010f5f?, 0x0?},
{0x0,
  ...})\n\tmath/big/natdiv.go:507 +0x34b\nmath/big.(*Int).Quo(0xc01dfc6840,
0xc01dfc6840,
  0xc017523800)\n\tmath/big/int.go:211
+0x78\ncosmossdk.io/math.LegacyDec.QuoIntMut({0x100016bf5?},
  {0x105b3bf18?})\n\tcosmossdk.io/math@v1.0.1/dec.go:410
+0x15b\ncosmossdk.io/math.LegacyDec.ImmutOpInt({0x10249dd80?},
  0x1032c54c8, {0xc01727e980?})\n\tcosmossdk.io/math@v1.0.1/dec.go:242
+0x162\ncosmossdk.io/math.LegacyDec.QuoInt(...)\n\tcosmossdk.io/math@v1.0.1/dec.go:4
06\ngithub.com/pryzm-finance/pryzm-
core/x/icstaking/keeper.Keeper.createRedelegationMsgs({{0x103890400,
  0xc0008d0fa0}, {0x103851688, 0xc001a9e630}, {0x103851688, 0x0}, {0xc0016cd350,
0x2c},
  {0x103878cf0, 0xc000df7960}, ...}, ...)\n\tgithub.com/pryzm-finance/pryzm-
core/x/icstaking/keeper/bridge_redelegate.go:129
  +0xb45\ngithub.com/pryzm-finance/pryzm-
core/x/icstaking/keeper.RedelegateBridge.redelegate({{{{_,
  _}, {_, _}}}, _}, {{0x103877588, 0xc016dc71d0}, {0x103890fd0, 0xc00d9fba00},
{{0xb,
  ...}, ...}, ...}, ...)\n\tgithub.com/pryzm-finance/pryzm-
core/x/icstaking/keeper/bridge_redelegate.go:32
  +0x118\ngithub.com/pryzm-finance/pryzm-
core/x/icstaking/keeper.msgServer.RebalanceDelegations({{{0x103890400,
  0xc0008d0fa0}, {0x103851688, 0xc001a9e630}, {0x103851688, 0x0}, {0xc0016cd350,
0x2c},
  {0x103878cf0, 0xc000df7960}, ...}}, ...)\n\tgithub.com/pryzm-finance/pryzm-
core/x/icstaking/keeper/msg_server_rebalance_delegations.go:27
  +0x61e\ngithub.com/pryzm-finance/pryzm-
core/x/icstaking/types._Msg_RebalanceDelegations_Handler.func1({0x103877588,
  0xc01a77eff0}, {0x1027f6d40?, 0xc0159dfb80})\n\tgithub.com/pryzm-finance/pryzm-
core/x/icstaking/types/tx.pb.go:1372
```

```
  +0x78\ngithub.com/cosmos/cosmos-sdk/baseapp.
(*MsgServiceRouter).RegisterService.func2.1({0x103878af8,
  0xc00d5cb8c0}, {0xc017527778?, 0x10000e80b?}, 0x102878800?,
0xc009d02348)\n\tgithub.com/cosmos/cosmos-
sdk@v0.47.2/baseapp/msg_service_router.go:113
  +0xd2\ngithub.com/pryzm-finance/pryzm-
core/x/icstaking/types._Msg_RebalanceDelegations_Handler({0x10287f640?,
  0xc0010f46c0}, {0x103878af8, 0xc00d5cb8c0}, 0x1032c65a8,
0xc01dfc62c0)\n\tgithub.com/pryzm-finance/pryzm-core/x/icstaking/types/tx.pb.go:1374
  +0x138\ngithub.com/cosmos/cosmos-sdk/baseapp.
(*MsgServiceRouter).RegisterService.func2({{0x103877588,
  0xc016dc71d0}, {0x103890fd0, 0xc00d9fba00}, {{0xb, 0x0}, {0xc01165d710, 0xa},
0x332d,
  {0x35f4fd0, ...}, ...}, ...}, ...)\n\tgithub.com/cosmos/cosmos-
sdk@v0.47.2/baseapp/msg_service_router.go:121
  +0x2e4\ngithub.com/cosmos/cosmos-sdk/baseapp.(*BaseApp).runMsgs(_, {{0x103877588,
  0xc016dc71d0}, {0x103890fd0, 0xc00d9fba00}, {{0xb, 0x0}, {0xc01165d710, 0xa},
0x332d,
  ...}, ...}, ...)\n\tgithub.com/cosmos/cosmos-sdk@v0.47.2/baseapp/baseapp.go:791
  +0x606\ngithub.com/cosmos/cosmos-sdk/baseapp.(*BaseApp).runTx(0xc000e474a0, 0x3,
  {0xc0045dad80, 0x110, 0x110})\n\tgithub.com/cosmos/cosmos-
sdk@v0.47.2/baseapp/baseapp.go:734
  +0xe25\ngithub.com/cosmos/cosmos-sdk/baseapp.(*BaseApp).DeliverTx(0xc000e474a0,
  {{0xc0045dad80?, 0x203003?, 0x203003?}})\n\tgithub.com/cosmos/cosmos-
sdk@v0.47.2/baseapp/abci.go:409
  +0x17a\ngithub.com/cometbft/cometbft/abci/client.
(*localClient).DeliverTxAsync(0xc0005b4360,
  {{0xc0045dad80?, 0x0?,
0x0?}})\n\tgithub.com/cometbft/cometbft@v0.37.1/abci/client/local_client.go:82
  +0x105\ngithub.com/cometbft/cometbft/proxy.
(*appConnConsensus).DeliverTxAsync(0xc0005c83c0,
  {{0xc0045dad80?, 0x20?,
0xb?}})\n\tgithub.com/cometbft/cometbft@v0.37.1/proxy/app_conn.go:106
  +0x102\ngithub.com/cometbft/cometbft/state.execBlockOnProxyApp({0x1038777f0?,
0xc0015548e0},
  {0x10388cd40, 0xc0005c83c0}, 0xc0005fd0e0, {0x103891d08, 0xc0005c8000},
0x332c?)\n\tgithub.com/cometbft/cometbft@v0.37.1/state/execution.go:376
  +0x812\ngithub.com/cometbft/cometbft/state.(*BlockExecutor).ApplyBlock(_, {{{0xb,
  0x0}, {0xc00154dee8, 0x6}}, {0xc00154df10, 0xa}, 0x1, 0x332c, {{0xc01a9997e0,
...},
  ...}, ...}, ...)\n\tgithub.com/cometbft/cometbft@v0.37.1/state/execution.go:197
  +0x151\ngithub.com/cometbft/cometbft/consensus.
(*State).finalizeCommit(0xc000045c00,
  0x332d)\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:1700
+0xaa5\ngithub.com/cometbft/cometbft/consensus.
(*State).tryFinalizeCommit(0xc000045c00,
  0x332d)\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:1609
+0x2ff\ngithub.com/cometbft/cometbft/consensus.
(*State).enterCommit.func1()\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state
.go:1544
  +0xaa\ngithub.com/cometbft/cometbft/consensus.(*State).enterCommit(0xc000045c00,
  0x332d, 0x0)\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:1582
+0xccf\ngithub.com/cometbft/cometbft/consensus.(*State).addVote(0xc000045c00,
  0xc00d68c960, {0x0,
0x0})\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:2212
  +0x1a30\ngithub.com/cometbft/cometbft/consensus.(*State).tryAddVote(0xc000045c00,
  0xc00d68c960, {0x0?,
0x1000b8826?})\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:2001
  +0x2c\ngithub.com/cometbft/cometbft/consensus.(*State).handleMsg(0xc000045c00,
{{0x103848240?,
  0xc016263b78?}, {0x0?,
0x0?}})\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:861
  +0x40b\ngithub.com/cometbft/cometbft/consensus.
(*State).receiveRoutine(0xc000045c00,
  0x0)\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:788
+0x505\ncreated
  by github.com/cometbft/cometbft/consensus.
(*State).OnStart\n\tgithub.com/cometbft/cometbft@v0.37.1/consensus/state.go:379
  +0x12d\n: panic"
```

The log shows a division by zero occurs because the total delegation amount is 0 at this moment.

Recommendation

Recommend adding an extra check to ensure the denominator is nonzero before performing division.

Alleviation

[Pryzm Team - 09/13/2023] :
The team heeded the advice and resolved the finding by adding the check to ensure the denominator is nonzero in the following commits:

- `a3f003be4dd0ede1e84ab49688f30d3d04e813af`
- `074a272bb61adb89b9a040ff9dadbc2634e572a8`
- `21b11ae1fab27b1befa9d6017de2ed6ffc499a31`
- `ca8db9e820e18435efaa9a98e3411d0fb7b38d5e`

KEK-01 | INCORRECT ACCOUNT NUMBER OF `tokenfactory` MODULE ACCOUNT

Category	Severity	Location	Status
Inconsistency	● Minor	keeper.go (374cad8): 79	● Resolved

Description

Files:

- `x/tokenfactory/keeper/keeper.go`
- `x/tokenfactory/keeper/genesis.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The tokenfactory module account is set in the `InitGenesis()` using the function `CreateModuleAccount()` :

x/tokenfactory/keeper/genesis.go

```
func (k Keeper) InitGenesis(ctx sdk.Context, genState types.GenesisState) {
    k.CreateModuleAccount(ctx)
...
```

x/tokenfactory/keeper/keeper.go

```
func (k Keeper) CreateModuleAccount(ctx sdk.Context) {
    moduleAcc := authtypes.NewEmptyModuleAccount(types.ModuleName, authtypes.
Minter, authtypes.Burner)
    k.accountKeeper.SetModuleAccount(ctx, moduleAcc)
}
```

However, the function `SetModuleAccount()` only sets the module account with the default account number 0, which creates an account with duplicated account number.

Reference:

- https://github.com/osmosis-labs/osmosis/pull/5534

Proof of Concept

To demonstrate the scenario, we fetch all the accounts via the command:

> prismd q auth accounts

Result:

```
- '@type': /cosmos.auth.v1beta1.BaseAccount
  account_number: "0"
  address: prism1y7zj229j9vvr99y4gcvjap8rmch6xvmdc0fgcn
  pub_key:
    '@type': /cosmos.crypto.secp256k1.PubKey
    key: AoQm+XrAaXGmDm9nh4wDAlVSz2HwalEH0QHKR6QeCzcD
  sequence: "1"
- '@type': /cosmos.auth.v1beta1.ModuleAccount
  base_account:
    account_number: "0"
    address: prism19ejy8n9qsectrf4semdp9cpknflld0j6na9j0y
    pub_key: null
    sequence: "0"
  name: tokenfactory
  permissions:
  - minter
  - burner
...
```

The result shows the tokenfactory module account has account number 0 and there is another account with 0 account number.

Recommendation

Recommend adopting the fix in this PR from Osmosis.

Alleviation

[Pryzm Team - 09/13/2023] :
The team heeded the advice and resolved the finding with the fix from Osmosis in the commit `2768e3012759eeacc53896d293ac29614acc8a02` .

KER-03 | LACK OF STATE VALIDATION FOR `WhitelistedRoute`

Category	Severity	Location	Status
Logical Issue	● Minor	order_execution.go (374cad8): 429~433; whitelisted_route.go (374cad8): 63	● Resolved

Description

Files:

- `x/amm/keeper/order_execution.go`
- `x/amm/keeper/whitelisted_route.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

There is no validation to ensure that the `whitelistedRoute` is enabled since the `whitelistedRoute` is possibly paused by calling the function `SetWhitelistedRouteEnabled` .

- x/amm/keeper/order_execution.go

```
429       route, found := k.GetWhitelistedRoute(ctx, tokenIn, tokenOut)
430       if !found {
431           err = fmt.Errorf("whitelisted route not found for pair %s-%s", tokenIn,
tokenOut)
432           return nil, err
433       }
```

- x/amm/keeper/whitelisted_route.go

```
// SetWhitelistedRouteEnabled sets a whitelistedRoute.Enabled
    func (k Keeper) SetWhitelistedRouteEnabled(
        ctx sdk.Context,
        tokenIn string,
        tokenOut string,
        enable bool,
    ) error {
        route, found := k.GetWhitelistedRoute(ctx, tokenIn, tokenOut)
        if !found {
            return sdkerrors.Wrapf(types.ErrWhitelistedRouteNotFound,
"whitelisted route not found for pair: %s, %s", tokenIn, tokenOut)
        }

        route.Enabled = enable
        return k.SetWhitelistedRoute(ctx, route)
    }
```

Recommendation

Consider adding an validation to ensure the `whitelistedRoute` is enabled.

Alleviation

[Pryzm Team - 09/15/2023] :

The team heeded the advice and resolved this issue in the commit `bf475653a4d3122e893ddc7c584562a086759bb1` .

POS-02 | POTENTIAL UNABLE TO ACQUIRE `token-in` TOKENS THAT HAVE NOT BEEN EXCHANGED

Category	Severity	Location	Status
Logical Issue	● Minor	x/flowtrade/keeper/position.go (flowtrade): 119, 135~137	● Resolved

Description

Files:

- `x/flowtrade/keeper/flow.go`

Commit:

- `930876154d4296a366ba2ca179c227c6663cc55b`

The function `ExitFlow` enables the owners of positions to exit the flow and retrieve the `token-in` tokens that haven't been exchanged. Nevertheless, if the flow has ended or the specified end time has elapsed, these position owners will no longer be able to recover these tokens that were not exchanged.

```
118     // return error if flow is not active
119     switch flow.Status {
120     case types.FlowStatus_ENDED:
121         return types.ErrFlowEnded
122     case types.FlowStatus_STOPPED:
123         return types.ErrFlowStopped
124     }
```

```
135     if flow.EndTime.Before(ctx.BlockTime()) {
136         return types.ErrFlowEnded
137     }
```

Recommendation

Considering the implementation of a function that would allow position owners to reclaim their un-exchanged tokens even after the flow has ended.

Alleviation

[Pryzm Team - 09/15/2023] :

The team heeded the advice and removed final state validation from exit flow method to resolve this issue in the commit

`1f3f76d316391b34131f6c7cb5178a349e61ade8` .

QUE-01 | INCOMPLETE INPUTS OF UNDELEGATION QUERY

Category	Severity	Location	Status
Volatile Code	● Minor	x/icstaking/client/cli/query_undelegation.go (pryzm-core-0c34472): 45~49, 57~59, 87~89	● Resolved

Description

Files:

- `x/icstaking/client/cli/query_undelegation.go`

Commit:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The query command `CmdShowUndelegation()` is used to query the undelegation of a host chain at a specific epoch, in which only the `HostChain` is specified.

```
55              argHostChain := args[0]
56
57              params := &types.QueryGetUndelegationRequest{
58                  HostChain: argHostChain,
59              }
```

However, the `QueryGetUndelegationRequest` also accepts the epoch.

```
type QueryGetUndelegationRequest struct {
    HostChain string `protobuf:"bytes,1,opt,name=host_chain,json=hostChain,proto3"
json:"host_chain,omitempty"`
    Epoch     uint64 `protobuf:"varint,2,opt,name=epoch,proto3"
json:"epoch,omitempty"`
}
```

Similarly, the query command `CmdListIncompleteUndelegation()` does not specify the `Pagination`.

```
 87             params := &types.QueryIncompleteUndelegationRequest{
 88                 HostChain: argHostChain,
 89             }
```

Recommendation

Recommend adding the inputs of the aforementioned query commands.

Alleviation

[Pryzm Team - 10/16/2023] :

The team heeded the advice and resolved the finding by adding the inputs in the query commands in the commit `2071270e0da84d91c6cc88ab226f0580b7785128` .

REF-01 | LACK OF VALIDATION OF THE `RefractableAsset.FeeRatios` FIELD

Category	Severity	Location	Status
Volatile Code	● Minor	refractable_asset.go (374cad8): 70	● Resolved

Description

Files

- `x/assets/types/refractable_asset.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The field `FeeRatios` in the `RefractableAsset` is not validated, which causes a new RefractableAsset whose FeeRatios are negative value or nil can be registered.

Recommendation

We recommend adding validation to ensure the `FeeRatios` filed has a valid value.

Alleviation

[Pryzm Team - 09/18/2023]: The team heeded the advice and resolved this issue in the commit `0c34472f03010ddc4048ba0727c33c6418d69c2a` .

TYP-02 | MISSING STATELESS CHECK OF `TransferChannel` IN MESSAGES

Category	Severity	Location	Status
Volatile Code	● Minor	message_instant_unstake.go (374cad8): 45; message_redeem_unstaked.go (374cad8): 45; message_stake.go (374cad8): 44; message_unstake.go (374cad8): 44	● Resolved

Description

Files:

- `x/icstaking/types/message_instant_unstake.go`
- `x/icstaking/types/message_redeem_unstaked.go`
- `x/icstaking/types/message_stake.go`
- `x/icstaking/types/message_unstake.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The stateless check of the linked messages are performed in the function `ValidateBasic()` , which misses the validation of the field `TransferChannel` .

Recommendation

Recommend adding an extra check to reject malformed messages.

Alleviation

[Pryzm Team - 09/13/2023] :
The team headed the advice and resolved the finding by adding the check of `TransferChannel` in the commit `aa9d594b97347afb07de9e5d283c8581683cc3af` .

VAU-01 | LACK OF MINIMUM LIQUIDITY RESTRICTION IN POOL INITIALIZATION

Category	Severity	Location	Status
Logical Issue	● Minor	vault_join.go (374cad8): 106	● Acknowledged

Description

Files:

- `x/amm/keeper/vault_join.go`
- `x/amm/keeper/pools/weightedmath/weighted_math_join.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

In the `amm` module, a pool is initialized with initial liquidity after creation. The initial liquidity is checked to ensure it is positive:

```
	if !lpOut.IsPositive() {
		return summary,
			sdkerrors.Wrap(types.ErrInvalidAmount,
"lpOut should be positive after initialization")
	}
```

However, a small liquidity value such as `1e-18` may lead to big numerical errors in further calculations. Therefore, the initial liquidity should be bounded by a minimum liquidity value.

The comment in `weighted_math_join.go` also indicates there should be a minimum liquidity value:

```
// and, because there is a minimum LPT, we round down the invariant.
```

It is also worth noting that the function `CheckSufficientLiquidity()` is applied to check minimum liquidity when users add liquidity to the pool:

```
// CheckSufficientLiquidity checks if balance is not less than one
func CheckSufficientLiquidity(balance sdk.Dec) error {
	if balance.LT(sdk.OneDec()) {
		return ErrInsufficientLiquidity
	}
	return nil
}
```

Proof of Concept

```
func (s *keeperTestSuite) TestInitializationWithLowLiquidity() {
    _, err := s.msgServer.CreateWeightedPool(s.ctx, &types.MsgCreateWeightedPool{
        SwapFeeRatio: sdk.MustNewDecFromStr("0.0001"),
        Creator:      s.authority,
        Tokens: []types.CreateWeightedPoolToken{
            {
                Denom:            "token1",
                NormalizedWeight: sdk.MustNewDecFromStr("0.5"),
            },
            {
                Denom:            "token2",
                NormalizedWeight: sdk.MustNewDecFromStr("0.5"),
            },
        },
        Name: "pool",
    })
    s.Require().NoError(err)

    tokens := s.ammKeeper.GetAllTokensForPool(s.ctx, 0)

    coin1 := sdk.NewCoin(tokens[0].Denom,
sdk.MustNewDecFromStr("10000000000000000").TruncateInt()) // 0.01
    coin2 := sdk.NewCoin(tokens[1].Denom,
sdk.MustNewDecFromStr("10000000000000000").TruncateInt()) // 0.01
    creatorStr := sample.AccAddress()
    creator := sdk.MustAccAddressFromBech32(creatorStr)

    s.bankKeeper.EXPECT().
        BlockedAddr(creator).
        Return(false)
    lpCoins := sdk.NewCoins(sdk.NewCoin("LP:0:pool",
sdk.MustNewDecFromStr("19999999999999599").TruncateInt())) // 0.02

    s.setupInitPoolMocks(creator, sdk.NewCoins(coin1, coin2), lpCoins)

    fmt.Println("Before msgServer.InitializePool.")
    _, err = s.msgServer.InitializePool(s.ctx, &types.MsgInitializePool{
        Creator:   creatorStr,
        PoolId:    0,
        AmountsIn: sdk.NewCoins(coin1, coin2),
    })
    s.Require().NoError(err)

    supply := s.ammKeeper.GetLpTokenSupplyOrZero(s.ctx, 0)
    s.Require().Equal(lpCoins[0].Amount, supply) // Supply is smaller than minimum
LPT.
}
```

Recommendation

Recommend checking the initial liquidity to ensure it is greater than one or another reasonable minimum liquidity value.

Alleviation

[Pryzm Team - 09/13/2023] :
We have a feature `initialization_allow_list` to limit the users that can initialize a pool, any pool sensitive to initialization can use this feature.

[CertiK - 11/20/2023] :
This finding describes the issue that the initial liquidity amount is not restricted by a lower boundary and it could lead to precision losses in further calculations.

Meanwhile, initialization_allow_list in the pool initialization is used to restrict accounts that can initialize (add liquidity to) the pool, but it does not restrict the initial liquidity amount.

WEI-02 | LACK OF CHECK FOR WEIGHT UPDATE PERIOD

Category	Severity	Location	Status
Logical Issue	● Minor	weighted_token.go (374cad8): 43	● Resolved

Description

Files:

- `x/amm/keeper/weighted_token.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

In the AMM module, pool creators are allowed to update the weights of tokens in pools gradually. The pool creators need to provide `startTimeUnixMillis` and `endTimeUnixMillis` so that the updates would happen within this time priod.

```
42
// UpdateWeightsGradually sets weights to update from current value to the given
values in the given time range

43  func (k Keeper) UpdateWeightsGradually(ctx sdk.Context, poolId uint64,
 startTimeUnixMillis int64,
44      endTimeUnixMillis int64, normalizedWeights map[string]sdk.Dec) error {
```

However, there is no check to ensure the minimum difference between `startTimeUnixMillis` and `endTimeUnixMillis` , which means the weight updates could be done within a short time period or even immediately instead of "gradually".

Proof of Concept

In the following test, weights are updated immediately.

```
func (s *keeperTestSuite) TestWeightUpdateImmediately() {
    s.setZeroProtocolFeeParams()
    s.createAndInitThreeTokenPool(0)

    poolData, found := s.ammKeeper.GetPool(s.ctx, 0)
    s.Require().True(found)
    api, err := pools.GetPoolApi(s.ctx, poolData, s.ammKeeper, nil, nil, nil)
    s.Require().NoError(err)

    weighted := api.(*pools.WeightedPool)
    weights, err := weighted.GetNormalizedWeights(s.ctx)
    s.Require().NoError(err)

    // weights should be as described in method s.createAndInitThreeTokenPool()
    s.Require().Equal([]types.TokenWeight{
        {
            Denom:            "token1",
            NormalizedWeight: sdk.MustNewDecFromStr("0.3"),
        }, {
            Denom:            "token2",
            NormalizedWeight: sdk.MustNewDecFromStr("0.2"),
        }, {
            Denom:            "token3",
            NormalizedWeight: sdk.MustNewDecFromStr("0.5"),
        },
    }, weights)

    // update weights again
    // current time is 11_500_000, which is used as StartTimeUnixMillis and
EndTimeUnixMillis so that the weights are updated immediately
    _, err = s.msgServer.UpdateWeights(s.ctx, &types.MsgUpdateWeights{
        Creator:             s.authority,
        PoolId:              0,
        StartTimeUnixMillis: 11_500_000,
        EndTimeUnixMillis:   11_500_000,
        TokenWeights: []types.TokenWeight{
            {
                Denom:            "token1",
                NormalizedWeight: sdk.MustNewDecFromStr("0.5"),
            },
            {
                Denom:            "token2",
                NormalizedWeight: sdk.MustNewDecFromStr("0.3"),
            },
            {
                Denom:            "token3",
                NormalizedWeight: sdk.MustNewDecFromStr("0.2"),
            },
        },
    })
    s.Require().NoError(err)

    timing, found := s.ammKeeper.GetWeightUpdateTiming(s.ctx, 0)
    s.Require().True(found)
    s.Require().Equal(types.WeightUpdateTiming{
        PoolId:          0,
        StartUnixMillis: 11_500_000,
        EndUnixMillis:   11_500_000,
    }, timing)

    tokens := s.ammKeeper.GetAllWeightedTokensForPool(s.ctx, 0)
    s.Require().Equal([]types.WeightedToken{
        {
            Denom:                 "token1",
            PoolId:                0,
            NormalizedStartWeight: sdk.MustNewDecFromStr("0.3"),
            NormalizedEndWeight:   sdk.MustNewDecFromStr("0.5"),
        },
        {
            Denom:                 "token2",
            PoolId:                0,
            NormalizedStartWeight: sdk.MustNewDecFromStr("0.2"),
            NormalizedEndWeight:   sdk.MustNewDecFromStr("0.3"),
        },
        {
            Denom:                 "token3",
            PoolId:                0,
            NormalizedStartWeight: sdk.MustNewDecFromStr("0.5"),
            NormalizedEndWeight:   sdk.MustNewDecFromStr("0.2"),
        },
    }, tokens)

    weights, err = weighted.GetNormalizedWeights(s.ctx)
    s.Require().NoError(err)

    // weights have been updated to new values
    s.Require().Equal([]types.TokenWeight{
        {
            Denom:            "token1",
            NormalizedWeight: sdk.MustNewDecFromStr("0.5"),
        }, {
            Denom:            "token2",
            NormalizedWeight: sdk.MustNewDecFromStr("0.3"),
        }, {
            Denom:            "token3",
            NormalizedWeight: sdk.MustNewDecFromStr("0.2"),
        },
    }, weights)
}
```

Results:

```
Running tool: /usr/local/go/bin/go test -timeout 30s -testify.m
^(TestWeightUpdateImmediately)$ github.com/pryzm-finance/pryzm-core/x/amm/keeper

WARNING: proto: file name query.proto does not start with expected testdata/; please
make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name testdata.proto does not start with expected testdata/;
please make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name tx.proto does not start with expected testdata/; please
make sure your folder structure matches the proto files fully-qualified names
WARNING: proto: file name unknonwnproto.proto does not start with expected
testdata/; please make sure your folder structure matches the proto files fully-
qualified names
PASS
ok      github.com/pryzm-finance/pryzm-core/x/amm/keeper    1.659s
```

Recommendation

Recommend adding a check for `startTimeUnixMillis` and `endTimeUnixMillis` to ensure the token weights are updated gradually.

Alleviation

[Pryzm Team - 09/13/2023] :

This works the same as the Balancer protocol and allows for immediate updating of weights. It allows for immediate weight update, which is also used in the token introduction process.

[CertiK - 11/20/2023] :

The function's name, "UpdateWeightsGradually", could potentially mislead users, given that it implies a gradual process, whereas weights might be updated instantly.

Furthermore, such immediate updates could influence the outcomes of user swaps. While the impact is mitigated since users can set minimum amounts to limit slippage, it would be more prudent for users to be informed about potential changes in the pool during their swap transactions

[Pryzm Team - 11/29/2023] :

The team heeded the advice and resolved the finding by implementing a minimum 7-day period for weight updates in the commit `b7003e637c1dba2370597ea5c71e721996636767` .

X0C-02 POTENTIAL KEY COLLISION BECAUSE DENOM COULD CONTAIN "/"

Category	Severity	Location	Status
Volatile Code	● Minor	x/amm/types/key_expiring_pool_token.go (pryzm-core-0c34472): 30~32; x/amm/types/key_pool_token.go (pryzm-core-0c34472): 30~32; x/amm/types/key_weighted_token.go (pryzm-core-0c34472): 19~21; x/assets/types/key_refractable_asset.go (pryzm-core-0c34472): 31~33; x/incentives/types/key_bond.go (pryzm-core-0c34472): 28~30; x/incentives/types/key_pool.go (pryzm-core-0c34472): 18~20	● Resolved

Description

Files:

- `x/amm/types/key_expiring_pool_token.go`
- `x/amm/types/key_weighted_token.go`
- `x/amm/types/key_pool_token.go`
- `x/assets/types/key_refractable_asset.go`
- `x/incentives/types/key_pool.go`
- `x/incentives/types/key_bond.go`

Commit:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The token denom in Cosmos SDK should match the regular expression:

```
    reDnmString = `[a-zA-Z][a-zA-Z0-9/:._-]{2,127}`
```

which implies the token denom could contain the "/".

In the aforementioned places, the "/" is appended to create the store key:

```
    denomBytes := []byte(denom)
    key = append(key, denomBytes...)
    key = append(key, []byte("/")...)
```

As a result, it could possibly lead to key collision in the future development.

Recommendation

Recommend using "|" instead of "/" to avoid potential store key collision.

Alleviation

[Pryzm Team - 10/16/2023] :
All of the mentioned files have "/" at the end of the keys. when a key ends with "/" we always know that the last "/" is a key separator and is not a part of the denom. Hence, this is not an issue for these methods. However, we found out we are using addresses as a part of the key for bonds, which might cause issues. Therefore we changed that to a length-prefixed address in the commit `d3ae58151679a2edd70c3d35833a556383956c55` .

[CertiK - 11/18/2023] :
Adding the length prefix of the address used for the bond key resolved the issue of the bond key. However, other keys that include the denom which ends with the "/" have not been handled. These places are flagged in the files of the description. Though the current design is not an issue, it could potentially lead to collision issues in future development. For example, suppose there are two prefixed stores with prefixes, `Bond/value/` and `Bond/` .

- If there are two denoms `XYZ` and `value/XYZ` , then it could possibly lead to a key collision because the "/" is a valid symbol in the denom that could be used in the second denom.

Considering the potential risk, the finding is marked as partially resolved. To fully resolve this issue, recommend using "|" instead of "/" for the keys containing the denom.

[Pryzm - 11/29/2023] :
The presented example for future implementation is flawed. Using key prefixes like "bond/" that overlap with others such as "bond/value" is inherently problematic. This should be avoided in all future implementations, independent of the denom issue.

ASS-02 | UNNECESSARY ARG IN THE `QueryGetMaturityLevelRequest`

Category	Severity	Location	Status
Coding Style	● Informational	query_maturity_level.go (374cad8): 68~70; grpc_query_maturity_level.go (374cad8): 80	● Resolved

Description

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

Files:

- `x/assets/keeper/grpc_query_maturity_level.go`
- `x/assets/client/cli/query_maturity_level.go`

To query the specified `MaturityLevel`, the query rpc service `MaturityLevel` and command `prismd q assets show-maturity-level` need the users to pass three args: `Active`, `AssetId`, and `Symbol`.

But the arg `Active` is unnecessary in the query method. The query method will get **ACTIVE** `MaturityLevel`s firstly. If any **ACTIVE** `MaturityLevel` exists, this query returns only the **ACTIVE** elements. If no **ACTIVE** `MaturityLevel` exists, the query method will return existing **DEACTIVATE** `MaturityLevel`s.

File: `x/assets/keeper/grpc_query_maturity_level.go`

```go
func (k Keeper) MaturityLevel(c context.Context, req *types.
QueryGetMaturityLevelRequest) (*types.QueryGetMaturityLevelResponse, error) {
    if req == nil {
        return nil, status.Error(codes.InvalidArgument, "invalid request")
    }
    ctx := sdk.UnwrapSDKContext(c)

    val, found := k.GetMaturityLevel(
        ctx,
        req.AssetId,
        req.Symbol,
    )
    if !found {
        return nil, status.Error(codes.NotFound, "not found")
    }

    return &types.QueryGetMaturityLevelResponse{MaturityLevel: val}, nil
}
```

File: `x/assets/keeper/maturity_level.go`

```go
func (k Keeper) GetMaturityLevel(ctx sdk.Context, asset string, symbol string)
(val types.MaturityLevel, found bool) {
    val, found = k.doGetMaturityLevel(ctx, true, asset, symbol)
    if found {
        return val, found
    }

    return k.doGetMaturityLevel(ctx, false, asset, symbol)
}
```

Since arg `Active` is required to query a specified `MaturityLevel`, and the actual query result is likely inconsistent with the input parameters, this can confuse the users.

Recommendation

We recommend removing the arg `Active` from the `QueryGetMaturityLevelRequest` OR modifying the logic in method `Keeper.MaturityLevel()`.

Alleviation

[Pryzm Team - 09/18/2023]:

The team heeded the advice and resolved this issue in the commit `0c34472f03010ddc4048ba0727c33c6418d69c2a`.

BAS-01 NO VALIDATION OF THE EXPIRING OR EXPIRED PASSET IN FUNCTION `JoinAllTokensGivenExactLptOut`

Category	Severity	Location	Status
Logical Issue	● Informational	x/amm/keeper/pools/base_weighted_pool.go (pryzm-core-17e20c2): 385, 443	● Resolved

Description

Files:

- `x/amm/keeper/pools/base_weighted_pool.go`

Commit:

- `17e20c2b046a1b389630270bfc10a1079ea0a177`

Users can utilize the `JoinTokenGivenExactLptOut` function to input a single asset and obtain the LP asset, ensuring that the expiring or expired pASSET is not permitted. However, when users employ the `JoinAllTokensGivenExactLptOut` function and input all assets in the pool to add the LP, the expiring or expired pASSET is allowed.

```
// ValidateJoinSingleToken prevents joins if token is expiring/introducing
func (yc *yammPoolController) ValidateJoinSingleToken(ctx sdk.Context,
token types.PoolToken) error {
    if yc.isTokenExpiringOrExpired(ctx, token) {
        return sdkerrors.Wrapf(types.ErrInvalidJoin,
"cannot join single for expiring token %s", token.Denom)
    }
    return nil
}
```

We would like to seek clarification from the Pryzm team to confirm if this behavior is as intended.

Recommendation

We recommend reviewing the logic again and ensuring it is as intended.

Alleviation

[Pryzm Team - 10/16/2023] :

When we want to remove a token from the pool, we should not allow increasing its balance, since it is desired to drain the token ASAP. That is why swaps, joinSingle, and joinExact operations are checked not to increase the balance of removing tokens. However, since we might have a removing token at any time, if we blocked proportional(all token) join/exit operations then it is likely to disable this feature at all! Keeping in mind that the balance of a removing token is not expected to be large, a proportional join would not increase it too much, and more importantly, we know that the design is not vulnerable to proportional increments in balances (since they do not change prices) so it is OK to allow for that.

FLO-01 | THE PURPOSE OF THE DEPOSIT `creationDeposit`

Category	Severity	Location	Status
Logical Issue	● Informational	x/flowtrade/keeper/flow.go (flowtrade): 54~60	● Resolved

Description

Files:

- `x/flowtrade/keeper/flow.go`

Commit:

- `930876154d4296a366ba2ca179c227c6663cc55b`

According to the provided logic, a user has the ability to send any amount of creation deposit, which is unrelated to the `token-out` amounts, to the module account in order to create a token flow. Additionally, the user can exit the flow and retrieve the creation deposit without any restrictions. The auditing team is seeking confirmation from the PRYZM team regarding whether this aligns with the intended design behavior. Furthermore, the team would like to know more details on the purpose and usage of the creation deposit.

```
    // send the creation deposit to the module account
    if creationDeposit != nil {
        err := k.sendCoinsFromAccountToModule(ctx, creator, sdk.NewCoins(*creationDeposit))
        if err != nil {
            return 0, err
        }
    }
```

Recommendation

Alleviation

[Pryzm Team - 07/19/2023]:
The creation deposit amount is set in the module params, and the flow creator must provide that amount in order to create a new flow. The purpose of this amount is to prevent spamming the system by creating too many flows.

The creation deposit is passed to the CreateFlow function, so other modules can create flows with or without this deposit (e.g. treasury module in prism-core creates flows with zero creation deposit). The deposit is loaded from params and passed to this method when the flow is created using the message server.

GEE-01 | MISSING VALIDATION OF `ChannelUndelegationList` IN ICSTAKING MODULE'S GENESIS STATE

Category	Severity	Location	Status
Volatile Code	● Informational	genesis.go (374cad8): 55	● Resolved

Description

Files:

- `x/icstaking/types/genesis.go`
- `x/icstaking/types/genesis.pb.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The icstaking module's genesis state is defined as follows:

x/icstaking/types/genesis.pb.go

```
// GenesisState defines the icstaking module's genesis state.
type GenesisState struct {
	Params                  Params
`protobuf:"bytes,1,opt,name=params,proto3" json:"params"`
	PortId                  string
`protobuf:"bytes,2,opt,name=port_id,json=portId,proto3" json:"port_id,omitempty"`
	HostChainList           []HostChain
`protobuf:"bytes,3,rep,name=host_chain_list,json=hostChainList,proto3"
json:"host_chain_list"`

	HostChainStateList      []HostChainState
`protobuf:"bytes,4,rep,name=host_chain_state_list,json=hostChainStateList,proto3"
json:"host_chain_state_list"`

	UndelegationList        []Undelegation
`protobuf:"bytes,5,rep,name=undelegation_list,json=undelegationList,proto3"
json:"undelegation_list"`

	ChannelUndelegationList []ChannelUndelegation
`protobuf:"bytes,6,rep,name=channel_undelegation_list,json=channelUndelegationList,p
roto3" json:"channel_undelegation_list"`

}
```

However, the `ChannelUndelegationList` is missing from both `DefaultGenesis()` and `Validate()`:

x/icstaking/types/genesis.go

```go
func DefaultGenesis() *GenesisState {
    return &GenesisState{
        PortId:             PortID,
        HostChainList:      []HostChain{},
        HostChainStateList: []HostChainState{},
        UndelegationList:   []Undelegation{},
        // this line is used by starport scaffolding # genesis/types/default
        Params: DefaultParams(),
    }
}
```

```go
func (gs GenesisState) Validate() error {
...
```

Recommendation

Recommend adding the `ChannelUndelegationList` to the `DefaultGenesis()` and validating it in `Validate()`.

Alleviation

[Pryzm Team - 09/13/2023] :

The team resolved the finding by adding the `ChannelUndelegationList` in the commit `61efe6b7eb418efac6de42a5c0d81823586a5dcc`.

GLOBAL-02 | COSMOS MESSAGES NEED TO EXTEND `cosmos.msg.v1.signer` OPTION

Category	Severity	Location	Status
Volatile Code	● Informational		● Resolved

Description

The issue #10933 arises from the implementation of the `sdk.Msg.GetSigners()` method, which is based on Golang. As a result, the information this method provides cannot be utilized by multi-chain dynamic clients, requiring manual filling of TX (transaction) authentication information. This limitation also extends to non-Golang language-based clients.

To address this concern and ensure compatibility, the cosmos sdk introduced a protobuf extension called `google.protobuf.MessageOptions` through PR #10977. To benefit from this extension, it is now required that all cosmos messages extend the `cosmos.msg.v1.signer` option.

For more detailed information about this change, please refer to the modification log in the following link: CHANGELOG.md.

Recommendation

Recommend making the necessary modifications to cosmos messages by extending the `cosmos.msg.v1.signer` option.

Alleviation

[Pryzm Team - 09/15/2023] :

The team resolved this issue in the commit `0c34472f03010ddc4048ba0727c33c6418d69c2a` .

GO3-01 | INSECURE COSMOS SDK VERSION

Category	Severity	Location	Status
Logical Issue	● Informational	go.mod (374cad8): 14	● Resolved

Description

The current prism-core implementation depends on Cosmos SDK `v0.47.2` :

```
github.com/cosmos/cosmos-sdk v0.47.2
```

However, recently, a high-severity issue (barberry security vulnerability) impacting cosmos-sdk was reported to core developers. This issue impacts chains running on v0.46 and v0.47, and chains on v0.45 may be vulnerable if they backport features of cosmos-sdk modules to their respective forks.

This issue has been resolved in Cosmos SDK `v0.47.3` , whose release notes suggest chains using Cosmos SDK `<= v0.47.2` to upgrade to `v0.47.3` immediately.

Recommendation

Recommend using Cosmos SDK `v0.47.3` instead of `v0.47.2` .

Alleviation

[Pryzm Team - 09/18/2023] :

The team heeded the advice and upgraded the Cosmos SDK to version v0.47.4 to resolve this issue in the commit `8b657530d2b4fb39689e93645a3ac0ae35e8554a` .

ICS-01 | TYPO IN MESSAGE AND FUNCTION NAME `RedeemInterchainAccount`

Category	Severity	Location	Status
Coding Style	● Informational	msg_server_redeem_interchain_account.go (374cad8): 12; tx.pb.go (374cad8): 886	● Resolved

Description

Files:

- `x/icstaking/keeper/msg_server_redeem_interchain_account.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

According to the logic, the message `MsgRedeemInterchainAccount` and function `RedeemInterchainAccount()` are intended to register the interchain accounts, which could be changed to `MsgRegisterInterchainAccount` and `RegisterInterchainAccount()`.

Recommendation

Recommend correcting the typo in relevant places.

Alleviation

[Pryzm Team - 09/13/2023] :

The team heeded the advice and resolved the finding by correcting the typo in the commit `f0ddd23ebbe5e19796b58b6af5e1fb7c2ddff7281`.

KED-01 DISCUSSION ON `ExchangeRate` UPDATING AND `YAsset` YIELD DISTRIBUTION

Category	Severity	Location	Status
Volatile Code	● Informational	x/refractor/keeper/keeper_distribution.go (pryzm-core-0c34472): 13	● Resolved

Description

Files:

- `x/refractor/keeper/keeper_distribution.go`

Commits:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The auditing team would like to check with the Pryzm team about two questions regarding the asset refractor.

Question 1: Implications of Refracting and Merging Assets in the `Refractor` Module

The `Refractor` has three primary functionalities: refracting assets (converting them to other assets), merging assets (combining assets), and redeeming assets. These operations involve a key data structure called `AssetState` , which keeps track of certain details for each refractable asset, including the asset's ID, the total amount of refracted `pAsset` , and the last known exchange rate.

```
// File: x/refractor/types/asset_state.pb.go, line 28
type AssetState struct {
    AssetId              string
    TotalPAmount         github_com_cosmos_cosmos_sdk_types.Int
    LastSeenExchangeRate github_com_cosmos_cosmos_sdk_types.Dec
}
```

One important insight is that when a user refracts some `cAsset` s, the number of `cAsset` s they can later merge or redeem from this module will always be less than the number of `cAsset` s refracted. This is because the conversion ratio is determined by `AssetState.TotalPAmount` divided by the related `cAsset` balance in the `Refractor` module. This ratio remains constant for refracting, merging, and redeeming transactions, and it can only change during yield distribution.

Yield distribution occurs when a new `ExchangeRate` is voted for in the `oracle` module, which is greater than the `AssetState.LastSeenExchangeRate` . The module then distributes `cAsset` s as rewards and fees, and the distribution amount is calculated as:

$$cAssetBalance - \frac{cAssetBalance \times LastSeenExchangeRate}{\text{new ExchangeRate}}$$

To illustrate this, let's consider a scenario:

1. There are 200 cLuna refracted, and the state for cLuna is {cLuna, 500, 2.5}, and 500 yAssets staked.
2. Alice uses 20 cLuna to refract, and Alice will get 50 `p:cLuna:31Dec2023` and 50 `y:cLuna:31Dec2023`; Alice stakes all of the yAssets. The `Refractor` module owes 220 cLuna, and the state updates to {cLuna, 550, 2.5}.
3. The `oracle` module gets a new ExchangeRate 2.0, and the state for cLuna updates to {cLuna, 550, 2.0}.
4. The `oracle` module gets a new ExchangeRate 2.5; 2.5 is greater than 2.0, so yields 44 cLuna will be distributed. Alice owes 50 yAssets in 550 yAssets, so Alice will receive 4 cLuna. The `Refractor` module owes 176 cLuna, and the state for cLuna updates to {cLuna, 550, 2.5}.
5. Alice merged 50 `p:cLuna:31Dec2023` and 50 `y:cLuna:31Dec2023`; the p:c ratio now is 550/176, so Alice can redeem 16 cLuna.

In this scenario, we can observe that even though Alice receives yields, assuming there is no protocol fee, Alice can only get 20 cLuna back. If there are fees or if Alice forgets to stake `yAsset`s or if Alice's `yAsset`s are expired, all `cAsset`s Alice can receive will be less than what she owned before. This highlights that users cannot profit from the `Refractor` and `ystaking` modules by refracting assets.

Question 2: Handling Unstaked `yAssets` and Protocol Fees

In the context of the project, if a `yAsset` is not staked in the `ystaking` module, all of the yields generated from those `yAssets` will be sent as protocol fees. This means that if users provide `yAssets` to the AMM (Automated Market Maker) pool and do not stake them, any yield generated from those `yAssets` will contribute to protocol fees rather than be distributed to users.

In summary, users must stake their `yAssets` in the `ystaking` module to receive yields since users can not get the profit from unstaked `yAssets`. Additionally, users should be aware that refracting `cAsset`s may result in reduced `cAsset` holdings over time due to yield distribution dynamics, which are influenced by fluctuations in exchange rates as determined by the 'oracle' module's voting mechanism.

Recommendation

Alleviation

[Pryzm Team - 10/17/2023]:

Exchange rate discussion

Question 1

The question provides the following scenario which we will discuss in detail:

> Let's consider a scenario:

> 1. There are 200 cLuna refracted, and the state for cLuna is {cLuna, 500, 2.5}, and 500 yAssets staked.
> 2. Alice uses 20 cLuna to refract, and Alice will get 50 `p:cLuna:31Dec2023` and 50 `y:cLuna:31Dec2023` ; Alice stakes all of the yAssets. The `Refractor` module owes 220 cLuna, and the state updates to {cLuna, 550, 2.5}.
> 3. The `oracle` module gets a new ExchangeRate 2.0, and the state for cLuna updates to {cLuna, 550, 2.0}.
> 4. The `oracle` module gets a new ExchangeRate 2.5; 2.5 is greater than 2.0, so yields 44 cLuna will be distributed. Alice owes 50 yAssets in 550 yAssets, so Alice will receive 4 cLuna. The `Refractor` module owes 176 cLuna, and the state for cLuna updates to {cLuna, 550, 2.5}.
> 5. Alice merged 50 `p:cLuna:31Dec2023` and 50 `y:cLuna:31Dec2023` ; the p:c ratio now is 550/176, so Alice can redeem 16 cLuna.
>
> In this scenario, we can observe that even though Alice receives yields, assuming there is no protocol fee, Alice can only get 20 cLuna back. If there are fees or if Alice forgets to stake `yAsset` s or if Alice's `yAsset` s are expired, all `cAsset` s Alice can receive will be less than what she owned before. This highlights that users cannot profit from the `Refractor` and `ystaking` modules by refracting assets.

First of all in order to provide a good context, we need to know how the exchange rate is computed and how it can change. The exchange rate reported through Oracle feeders or the icStaking module is computed according to the staked amount of underlying assets and the total supply of cASSET.

Suppose we have `1000` Luna tokens staked on terra, with a cLuna supply of `400` . This means the current exchange rate is `2.5` . This exchange rate is not changed by staking/unstaking operations. It is changed only if we have rewards or slash for the staked assets. Hence the only way of changing the exchange rate to `2` is when we get slashed for 200 Luna, then we would have 800 Luna staked and 400 cLuna resulting in an exchange rate of `2` . This can only get back to `2.5` as a result of staking rewards!

Our protocol defines yASSET and pASSET as follows:

- pASSET represents the principal, which means it represents the underlying assets staked on a staking system.
- yASSET represents the yields to be accrued for an amount of staked assets.

Now let's check the changes in the exchange rate again. The exchange rate decreases when we get slashed.

- **Decreasing exchange rate:** In a slash event the staked assets are being affected, meaning the underlying staked asset is decreased. Given our definitions of pASSET we can understand this event should affect pASSET holders and not the yASSET holders. This means if we get slashed, a p/y holder should get less out of redeem/merge actions. PLEASE note that slashing also affects the yield in a sense, when the total staked amount is decreased then the yield accrued would be less than before which would be automatically reflected to yASSET holders.
- **Increasing exchange rate:** When a reward is accrued, it is obvious that the yASSET holder should benefit from this reward and not the pASSET holder. Therefore, when the exchange rate increases, we should distribute yield to yASSET holders (staked yASSET).

Now, let's go over the example from the question again.
Initial state:

- **staking contract (or icStaking):** 1000Luna staked, and 400cLuna minted.
- **refractor:** 200cLuna in the vault and 500pLuna minted.
- **exchange rate:** `2.5`
- `Alice` is holding 20cLuna, corresponding to 50Luna staked.

Actions:

1. Alice refracts 20cLuna, and gets 50pLuna, 50yLuna
   a. This changes the refractor to 220cLuna in the vault and 550pLuna minted.
2. Alice stakes 50yLuna
3. The exchange rate goes to `2`, so we've had a 20% slash equivalent to 200Luna.
   a. This means the staking contract state is: 800Luna and 400cLuna.
4. The exchange rate is updated to `2.5`, so we've had a 25% reward equivalent to 200Luna. a. This results in reward distribution, and Alice gets 4cLuna as a reward.

At this stage Alice has received 4cLuna (=10Luna) as her exact share of the accrued reward, however, her p/y asset can only redeem 16cLuna (=40Luna). This means the yASSET holder has gained the rewards as explained earlier, while the pASSET holder (assuming maturity) has lost 10Luna because of slashing. This is exactly how the protocol needs to work.

In another sense, what PRYZM does is refracting principal from yield and allowing for trading either of these separately. Therefore, if a user refracts the cASSET and stakes the yASSET while holding the pASSET, they are not going to gain more compared to not refracting at all. This is the case for the given example, if Alice does not refract her cASSET, she is still being slashed and rewarded the same as before. NOTE: we do have numerous ways of generating yield in the system like liquidity pools where you can gain profit by providing the pASSET as liquidity. But here we are only discussing the refractor functionality in isolation.

It is important to also note that slashing is not a common thing and is always nothing compared to reward amounts, meaning that in the real world, Alice is always going to gain profit by just holding cASSET or refracting and staking the yASSET.

Conclusion

So far, we have explained how the Refractor module effectively separates the principal from the yield, incorporating a design that takes into account both slashing and rewards. In this design, the pASSET is subject to slashing due to its inherent nature as a representation of the principal, while the yASSET is subject to rewards as the yield token. It is important to note that slashing is not a side-effect of PRYZM and you might be slashed by just staking your assets anywhere (Although in a real-world scenario, the slashing is not substantial).

The refractor module works like an entry gate to the PRYZM protocol, in the protocol we have various features enabling p/y and c token holders to gain profit. For example, if you are an active trader you can trade these assets when you think their value is going to increase/decrease so you gain profit from trades, as well as doing arbitrage in the system. You can also stake your yASSETs to gain the staking yield and provide p/c tokens in liquidity pools to gain swap fees. We even have the ability to provide multiple levels of yields by providing liquidity pools for LP tokens or using Incentives and Alliance modules.

Question 2

This question is focused on unstaked yASSET:

> In the context of the project, if a `yAsset` is not staked in the `ystaking` module, all of the yields generated from those `yAssets` will be sent as protocol fees. This means that if users provide `yAssets` to the AMM (Automated Market Maker) pool and do not stake them, any yield generated from those `yAssets` will contribute to protocol fees rather than be distributed to users.
>
> In summary, users must stake their `yAssets` in the `ystaking` module to receive yields since users can not get the profit from unstaked `yAssets`. Additionally, users should be aware that refracting `cAsset`s may result in reduced `cAsset` holdings over time due to yield distribution dynamics, which are influenced by fluctuations in exchange rates as determined by the 'oracle' module's voting mechanism.

We have already explained how Refractor and Oracle work, so the only thing to mention here is what happens to unstaked yASSET. The first thing to mention is: that we do not have yASSET liquidity pools in AMM, so users are not going to provide yASSET as liquidity in AMM. Hence, a user is either trading a yASSET using amm (which is not supposed to take too long) or they have staked their yASSET. Other users holding yASSET with no aim, are not gaining profit which makes sense. In fact, if we want to track all yASSET holders, it would be too complex to implement and would not make sense, since users can simply stake their yASSET instead of holding it. Note that we do not have unbonding period in yStaking so users can stake and unstake their yASSET seamlessly.

There is only one feature in AMM that can lock yASSET, which is Pulse-Trade. This is a functionality for trading yASSETs over a long period of time, however, the Pulse-Trade is designed not to minimize the locked amount. Pulse-Trade only locks the required amount for executing one single step of the long-term multi-step trade. Therefore this is not a problem again for our system.

KEE-08 INCONSISTENT FUNCTION NAME `NewRedelegateMessageBridge()`

Category	Severity	Location	Status
Coding Style	● Informational	bridge_redelegate.go (374cad8): 24; keeper.go (374cad8): 91	● Resolved

Description

Files:

- `x/icstaking/keeper/keeper.go`
- `x/icstaking/keeper/bridge_redelegate.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

In order to be consistent with other bridge function names, the function `NewRedelegateMessageBridge()` could be changed to `NewRedelegateBridge()` .

Recommendation

Recommend changing the aforementioned function name.

Alleviation

[Pryzm Team - 09/13/2023] :

The team heeded the advice and resolved the finding by changing the function name in the commit `21b11ae1fab27b1befa9d6017de2ed6ffc499a31` .

KEP-01 | DISCUSSION ON PRICES OF `token-in` AND `token-out`

Category	Severity	Location	Status
Logical Issue	● Informational	x/flowtrade/keeper/flow.go (flowtrade): 14; x/flowtrade/keeper/position.go (flowtrade): 12	● Resolved

Description

Files:

- `x/flowtrade/keeper/flow.go`
- `x/flowtrade/keeper/position.go`

Commit:

- `930876154d4296a366ba2ca179c227c6663cc55b`

A User has the ability to create a flow and sell tokens by locking a deposit, while other users can join the flow by providing a certain amount of tokens. The tokens being sold, which are provided by the flow creator, are referred to as `token-out` . On the other hand, the tokens provided by the buyers are called `token-in` . And there is no restriction on the token amount of `token-in` and `token-out` .

Hence, the price of tokens is determined by considering the overall quantity of each token involved. This means that it is possible for a user to spend only a small number of tokens while acquiring a significantly larger amount of other tokens.

The auditing team would like to confirm this behavior aligns with the intended design.

Recommendation

Alleviation

[Pryzm Team - 07/19/2023]:
The purpose of `flowtrade` module is to provide an open market in which the price is determined by the amount of supply and demand, token-out, and token-in. The module enforces a minimum duration for each flow (set by governance in module parameters) to make sure that there is enough time for anyone to participate.

[CertiK - 11/18/2023]:
After review, the auditing team confirmed this behavior aligns with the intended design, so the discussion is marked as Resolved.

[Pryzm Team - 11/29/2023]:
The team introduced the new `LimitPrice` feature in `Flow` to mitigate token loss under specific conditions. Additionally,

they incorporated the `ExitWindowDuration` in `Flow` to provide enhanced protection for participants joining `Flow`. The changes were reflected in the commit 4e7068c78b18e62935ae3718c1b7c53fe00a1674.

MES-01 | MISSING VALIDATION OF `epoch` IN MESSAGE `MsgRedeemUnstaked`

Category	Severity	Location	Status
Volatile Code	● Informational	message_redeem_unstaked.go (374cad8): 45	● Resolved

Description

Files:

- `x/icstaking/types/message_redeem_unstaked.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The following `ValidateBasic()` is used to validate the message `MsgRedeemUnstaked` , which misses the validation of `epoch` .

```
45  func (msg *MsgRedeemUnstaked) ValidateBasic() error {
46      if _, err := sdk.AccAddressFromBech32(msg.Creator); err != nil {
47          return sdkerrors.Wrapf(errortypes.ErrInvalidAddress,
"invalid creator address (%s)", err)
48      }
49      if len(msg.HostChain) == 0 {
50          return sdkerrors.Wrapf(ErrHostChainNotFound, "host chain key is empty")
51      }
52      if msg.UAmount.IsNil() || msg.UAmount.LTE(sdk.ZeroInt()) {
53          return sdkerrors.Wrapf(ErrInvalidAmount, "u amount %d must be positive"
, msg.UAmount)
54      }
55      return nil
56  }
```

As a result, a zero epoch can be passed. However, the minimum epoch is 1 according to the implementation:

x/icstaking/keeper/epoch_counter.go

```go
func (k Keeper) GetCurrentUndelegationEpoch(ctx sdk.Context, hostChainId string
) uint64 {
    store := prefix.NewStore(ctx.KVStore(k.storeKey), types.KeyPrefix(types.
UndelegationEpochKeyPrefix))
    b := store.Get([]byte(hostChainId))
    if b == nil {
        return 1
    }
    value := &gogotypes.UInt64Value{}
    k.cdc.MustUnmarshal(b, value)
    return value.Value
}
```

Note that missing validation of `TransferChannel` has been pointed out in another finding.

Recommendation

Recommend adding an extra check to ensure the epoch is positive.

Alleviation

[Pryzm Team - 09/13/2023] :

The team heeded the advice and resolved the finding by adding the extra check to ensure epoch is positive in the commit `aa9d594b97347afb07de9e5d283c8581683cc3af1` .

MIN-01 | DISCUSSION ON THE CALCULATION OF THE MINTED TOKEN

Category	Severity	Location	Status
Incorrect Calculation	● Informational	hooks.go (374cad8): 38; minter.go (374cad8): 37~42	● Resolved

Description

Files:

- `x/mint/keeper/hooks.go`
- `x/mint/keeper/minter.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The `mint` module is responsible for minting tokens and distributing them to various modules, including the `Incentive` module, at the end of each epoch. The `Incentive` module receives these rewards and further distributes them to its associated pools. From this, we can infer that there is a connection between the `mint` and `Incentive` modules. When the `mint` module calculates the amount of tokens to be minted, it should consider the `bondRatio` specified in the `Incentive` module, rather than the `bondRatio` in the `staking` module.

In addition, the auditing team would like to inquire whether the Pryzm team has any whitepaper or document that can provide an explanation of the inflation formula: `(1 - bondedRatio/GoalBonded) * InflationRateChange`.

```
37      inflationRateChangePerYear := sdk.OneDec().
38          Sub(bondedRatio.Quo(params.GoalBonded)).
39          Mul(params.InflationRateChange)
40
41      inflationRateChange := inflationRateChangePerYear.Quo(sdk.NewDec(
epochsPerYear))
```

Recommendation

Alleviation

[Pryzm Team - 07/05/2023] :

The BondedRatio in the staking module is used to calculate the percentage of PRYZM tokens that are currently being used to secure the network through the staking process. On the other hand, the incentives module does not use the concept of BondedRatio. This module is designed to incentivize non-PRYZM token holders who are participating in the PRYZM network through various means, such as providing liquidity, voting on proposals, or participating in other community-driven initiatives. In short, the incentives module encourages participation in the PRYZM ecosystem by offering rewards or bonuses to users who contribute in these ways.

The mentioned formula is an exact copy of the default inflation calculation in cosmos-sdk mint module. Here is a link to the documentation.

[CertiK - 11/18/2023]:
After review, the auditing team confirmed the implementation meets the intended design, so the discussion is marked as Resolved.

MSG-02 EQUALITY COULD POSSIBLY NOT BE SATISFIED DUE TO ROUNDING ISSUE

Category	Severity	Location	Status
Volatile Code	● Informational	msg_server_redeem_unstaked.go (374cad8): 51	● Resolved

Description

Files:

- `x/icstaking/keeper/msg_server_redeem_unstaked.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

According to the calculation of variable `redemptionRate` :

```
45
// calculate the amount to be redeemed to user, based on the redemption rate of the
undelegation

46      redemptionRate := sdk.NewDecFromInt(undelegation.ReceivedAmount).QuoInt(
undelegation.TotalCAmount)
47      amount := redemptionRate.MulInt(msg.UAmount).TruncateInt()
```

The following equality could possibly never be satisfied due to the rounding error.

```
49      // add the amount to the claimed assets of channel undelegation record
50      undelegation.ClaimedAmount = undelegation.ClaimedAmount.Add(amount)
51      if undelegation.ClaimedAmount.Equal(undelegation.ReceivedAmount) {
```

Recommendation

The auditing team would like to confirm with the Pryzm team if it is the intended design.

Alleviation

[Pryzm Team - 09/13/2023] :

The team resolved the finding by replacing `ClaimedAmount` with `ClaimedUAmount` in the commit `7547eff3d251810c4a8b7757bbf28394bc473bdf` .

ORA-01 | POSSIBLE INCREASE OF EXCHANGE RATE

Category	Severity	Location	Status
Logical Issue	● Informational	oracle_callback.go (374cad8): 89~94	● Acknowledged

Description

Files:

- `x/icstaking/keeper/oracle_callback.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The exchange rate is updated in the function `OnMajorityVote()` according to the following formula:

$$\text{hostChainState.ExchangeRate} = \frac{\text{total delegation} + \text{delegation queue} + \text{hostChainState.AmountToBeCompounded} + \text{hostChainState.AmountToBeDelegated}}{\text{totalCTokenSupply}}$$

```
86      totalCTokenSupply := c.k.bankKeeper.GetSupply(ctx, hostChain.CDenom()).
Amount
87      if !totalCTokenSupply.IsZero() {
88          oldER := hostChainState.ExchangeRate
89          hostChainState.ExchangeRate = sdk.NewDecFromInt(
90              totalDelegation.
91                  Add(delegationQueueAmount).
92                  Add(hostChainState.AmountToBeCompounded).
93                  Add(hostChainState.AmountToBeDelegated)).
94              QuoInt(totalCTokenSupply)
95          err := c.k.exchangeRateListeners.ExchangeRateUpdated(ctx, hostChainId,
&oldER, hostChainState.ExchangeRate)
96          if err != nil {
97              return err
98          }
99      }
```

in which the `hostChainState.AmountToBeCompounded` is assigned to the balance of the reward account:

```
79      hostChainState.AmountToBeCompounded = payload.RewardAccountBalance
```

Since the reward account is able to receive coins from users, then the exchange rate could be increased if some users send coins to the reward account.

Recommendation

The auditing team would like to confirm with the Pryzm team if this scenario has been taken into account.

Alleviation

[Pryzm Team - 08/24/2023] :

The reward account's balance determines the amount of tokens that are rewarded to the stakers. If a user sends tokens to this account, it is counted as a reward and both stakers and protocol will take profit.

[CertiK - 11/18/2023] :

The issue has been designated as an Informational finding and has been marked as Acknowledged. While not a significant concern, it is essential to inform the team of the existence of this scenario.

PAR-01 | TYPO IN ERROR MESSAGES

Category	Severity	Location	Status
Coding Style	● Informational	params.go (374cad8): 239, 262	● Resolved

Description

Files:

- `x/icstaking/types/params.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

There are some typos in the error message:

x/icstaking/types/params.go

- in line 239, `DelegationInterval %d is not in range [0, 24] hours` should be `UndelegationInterval %d is not in range [0, 120] hours`
- in line 262, `max messages %d is not in range [1, 10] hours` should be `max messages %d is not in range [1, 10]`

Recommendation

Recommend correcting the typos to improve the code readability.

Alleviation

[Pryzm Team - 09/13/2023] :
The team heeded the advice and resolved the finding by correcting the typos in the commit `e64eec01b7059ce37542afbf72d3e26644e329b0` .

PRY-01 GAS IS NOT CONSUMED IF AN ERROR OCCURS BEFOREHAND

Category	Severity	Location	Status
Logical Issue	● Informational	x/amm/keeper/pool.go (pryzm-core-0c34472): 24; x/amm/keeper/vault_batch_swap.go (pryzm-core-0c34472): 129; x/amm/keeper/vault_exit.go (pryzm-core-0c34472): 47, 119, 194, 253; x/amm/keeper/vault_join.go (pryzm-core-0c34472): 34, 224, 301, 363; x/amm/keeper/vault_swap.go (pryzm-core-0c34472): 41; x/amm/keeper/order.go (pryzm-core-374cad8): 51	● Acknowledged

Description

Files:

- `x/amm/keeper/pool.go`
- `x/amm/keeper/vault_batch_swap.go`
- `x/amm/keeper/vault_exit.go`
- `x/amm/keeper/vault_join.go`
- `x/amm/keeper/vault_swap.go`
- `x/amm/keeper/order.go`

Commit:

- `0c34472f03010ddc4048ba0727c33c6418d69c2a`

The `ConsumeGas()` function is used in the Pryzm's amm module to increase the GasMeter's consumed gas by a predefined fixed amount, which varies to different message execution. This approach could be inappropriate as gas is only consumed after a successful execution or in the middle of these functions. In the case that the function returns an error before the execution of `ConsumeGas()` , the GasMeter will not be increased so the gas will not be consumed. However, the computation has already been performed and gas is supposed to be consumed accordingly.

The function `ConsumeGas()` that consumes the flat gas is called in the following functions and lines:

- x/amm/keeper/order.go:30
- x/amm/keeper/pool.go:24
- x/amm/keeper/vault_batch_swap.go:129
- x/amm/keeper/vault_exit.go:47
- x/amm/keeper/vault_exit.go:119
- x/amm/keeper/vault_exit.go:194

- x/amm/keeper/vault_exit.go:253
- x/amm/keeper/vault_join.go:34
- x/amm/keeper/vault_join.go:224
- x/amm/keeper/vault_join.go:301
- x/amm/keeper/vault_join.go:363
- x/amm/keeper/vault_swap.go:41

Recommendation

The auditing team would like to confirm with the Pryzm team if the aforementioned scenario has been taken into consideration. Otherwise, recommend moving the aforementioned `ConsumeGas()` calls to the beginning of the functions to ensure sufficient gas is consumed.

Alleviation

[Pryzm Team - 10/16/2023] :
Order, Pool, join, exit, swap: a set of read operations are called, and the store implementation consumes sufficient gas for these operations. Then the constant gas is consumed exactly before doing heavy math computations. This is in fact like all other methods where a tx fails before doing a read/write on a store, so gas for the store access is not consumed if input is not valid. Batch swap: for each step we consume gas and then execute the step, so if the first step fails, there is no need to consume gas for other steps of a batch.

[CertiK - 11/20/2023] :
Following the confirmation from the Pryzm team that this is an intended design choice, the finding has been classified as Informational and updated to Acknowledged status.

TOK-01 INCORRECT ERROR MESSAGE IN THE VALIDATION OF `CircuitBreakerSettings`

Category	Severity	Location	Status
Coding Style	● Informational	token_circuit_breaker_settings.go (374cad8): 49	● Resolved

Description

Files:

- `x/amm/types/token_circuit_breaker_settings.go`

Commit:

- `374cad8c0f54b4f98efb6248cf434524bf2b7f65`

The function `Validate()` is used to validate the `CircuitBreakerSettings` , in which the field `UpperBound` is validated as follows:

```
  47      if !m.UpperBound.IsZero() &&
  48          (m.UpperBound.GT(sdk.MustNewDecFromStr(maxNonZeroBound)) || m.
UpperBound.LT(m.LowerBound)) {
  49          return sdkerrors.Wrapf(ErrInvalidCircuitBreakerSettings,
"circuit-breaker UpperBound should be less than or equal to %s, and not less than
LowerBound"
, minNonZeroBound)
  50      }
```

This check ensures the `UpperBound` does not exceed `maxNonZeroBound` , which means the `minNonZeroBound` should be `maxNonZeroBound` in the error message.

Recommendation

Recommend changing `minNonZeroBound` to `maxNonZeroBound` in the aforementioned error message.

Alleviation

[Pryzm Team - 09/18/2023] :

The team heeded the advice and resolved this issue in the commit `200135e692be2fe9d15bbeab5c1437df53e38f1f` .

APPENDIX | PRYZM

Finding Categories

Categories	Description
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Incorrect Calculation	Incorrect Calculation findings are about issues in numeric computation such as rounding errors, overflows, out-of-bounds and any computation that is not intended.
Denial of Service	Denial of Service findings indicate that an attacker may prevent the program from operating correctly or responding to legitimate requests.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

DISCLAIMER | CERTIK

UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

CertiK | **Securing** the **Web3** World

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.