M^0 Labs Engineering, March 2024

Audits reports and fixes overview

[172 total issues and suggestions addressed]

Three Sigma [32 total, report]

Issue	Severity [Declared, Suggested]	Description & Notes	Fix PR / commit	Owner	Status	Duplicate of
C01 #2 Validator signature with zero timestamp can always be replayed	Critical Medium	No check for timestamp != 0 in `updateCollateral`	PR: https://github.com/MZero-Labs/protocol/pull/114 Commit in main: https://github.com/MZero-Labs/protocol/commit/b2c421c132cf6af6a18860ad17285b900be83163	Antonina Norair Michael De Luca	Resolved	https://gist.github.com/0xmonsoon/447ab8db4e80bbbbab2b93d25f1b2a6d
H01 #25 MToken's total principal invariant can be broken	High Medium ?	unchecked total earning principal addition in `MToken._addEarningAmount`	PR: https://github.com/MZero-Labs/protocol/pull/126 Commit in main: https://github.com/MZero-Labs/protocol/commit/74523a8e77be8654b902baaca41201d135c72190 Since this PR has been superseded, probably better to show the diff to later version of main	Pierrick Turelier	Resolved	https://github.com/threesigmaxyz/mzero-labs-8-1-2024-issues-external/issues/23
H02 #21 Lack of deadline in	High Low	Add expiration time for POWER buy order	PR: https://github.com/MZero-Labs/ttg/pull/216	Antonina Norair	Resolved	

PowerToken.buy can lead to user's cashToken being distributed through ZeroToken holders			Commit in main: https://github.com/MZero-Labs/ttg/commit/3fb74e72f03d45e7ec56f5c420143bcb627ac206			
M01 #27 Creating a new proposal in StandardGovernor may reach a state of permanent DoS	Medium Low	Power target supply overflow, missing cast	PR: https://github.com/MZero-Labs/ttg/pull/217 Commit in main: https://github.com/MZero-Labs/ttg/commit/c2131a5fb1dc1d8f4eb12874165016b90885db2d	Michael De Luca Pierrick Turelier	Resolved	
M02 Validator signatures with greater timestamps can be reused in a subsequent updateCollateral #5	Medium Low? Design?	No time boundaries for the set of validator signatures Reply: https://github.com/threesigmaxyz/mzero-labs-8-1-2024-issues-external/issues/5#issuecomment-1944027334		Antonina Norair	Won't fix	
M03 #1 A sufficiently large collateral may break the maximum owed M calculation	Medium Low	Multiplication within unchecked block can lead to overflow	PR: https://github.com/MZero-Labs/protocol/pull/117 Commit in main: https://github.com/MZero-Labs/protocol/commit/06b6cb1593da5baffa60d50fb75a0767dd754a6b	Michael De Luca	Resolved	
L01 #33 Multiplication after division in StableEarnerRateModel leads to loss of precision	Low Low	Unnecessary * 1e12 / 1e12 inside EarnerRateModel calculations	PR: https://github.com/MZero-Labs/protocol/pull/124 Commit in main: https://github.com/MZero-Labs/protocol/commit/cac3c24b038d9d6c71bad1132b9f4617e91ef2f1	Michael De Luca Antonina Norair	Resolved	

L02 #29 A decrease in updateCollateralInterval will lead to unfair penalties	Low Informational	A decrease in *updateCollateralInterval* can lead to unfair penalization of minters.	Explanation: https://github.com/threesigmaxyz/mzero-labs-8-1-2024-issues-external/issues/29#issuecomment-1928071298	Antonina Norair	Won't fix. Explanation added	
L03 #28 Unhandled rounding error in DistributionVault.getClaimable leads to locked dust bug	Low Informational	Dust/lost token is inevitable in any system since divisions are unavoidable. Dust could be lost by sending to the wrong/incompatible address as well. The best solution to somewhat mitigate dust is to accumulate distributions with scaling over various epochs, and then divide by the scale. Effectively, the more epochs one claims over, the more "dust" will be added together to form whole units that can be claimed. See the PR.	PR: https://github.com/MZero-Labs/ttg/pull/234 Commit in main: https://github.com/MZero-Labs/ttg/commit/3ae0bd3eda9d0399074549bdd9cdfe5e2f064ebf	Michael De Luca	Resolved	
L04 #23 MToken's total principal invariant doesn't hold without MinterGateway, leading to potential principal loss	Low	unchecked addition of *principalOfTotalEarningSupply* in *_addEarningAmount*	PR: https://github.com/MZero-Labs/protocol/pull/126 Commit in main: https://github.com/MZero-Labs/protocol/commit/74523a8e77be8654b902baaca41201d135c72190 Since this PR was superseded, perhaps a diff to later version of main is better	Pierrick Turelier	Resolved	https://github.com/threesigmaxyz/mzero-labs-8-1-2024-issues-external/issues/25
L05 Unrealized inflation calculation returns wrong value when balance reaches cap #22	Low	Unrealized inflation is maxed to `type(uint240).max` instead of balance	PR: https://github.com/MZero-Labs/ttg/pull/233/files#diff-767e30400d380a58c1330b6a340ea2b13ffb8544ace23c95f0335e6fde3fbfbdR266-R286 Commit in main: https://github.com/MZero-Labs/ttg/commit/0c86903456266e3814d5086c63b680063d03bdb6	Michael De Luca Antonina Norair	Resolved	Quantstamp H01 ChainSecurity CS-MZEROCORE-014 Independent auditor L03

L06 #20 Custom error for overflowing the total principal is not raised	Low Informational	Casting to *uint256* is missed in addition, it silently overflows *uint112*. Informational seems more appropriate since it would still revert but the custom error wouldn't be raised.	PR: https://github.com/MZero-Labs/protocol/pull/126 Commit in main: https://github.com/MZero-Labs/protocol/commit/bdd94b9952952c525f9e91ccbd762af792da7d4b	Pierrick Turelier	Resolved	Certora L01
L07 #16 Proposals in the same voting period can have different ids but do the same	Low	Calldata can have appended "dust" bytes in the end, so it cannot be used for proposalId generation without truncation	PR: https://github.com/MZero-Labs/ttg/pull/218 Commit in main: https://github.com/MZero-Labs/ttg/commit/d253ae1d01466aad691f2618513c6de642c1d9d9	Antonina Norair	Resolved	
L08 #4 Function `cancelMint` can be frontrunned to grief a validator	Low Informational	Potential gas griefing of validator by fraudulent minter by submitting the same mint request before they are canceled	Explanation: https://github.com/threesigmaxyz/mzero-labs-8-1-2024-issues-external/issues/4#issuecomment-1928050933	Antonina Norair	Won't fix. Explanation added	
L09 #3 StandardGovernor's implementation of quorum is incompatible with Tally	Low informational		While not ideal, the best we can do is to return 1 for quorum, which would mean "at least 1 for vote is enough to meet quorum" PR: https://github.com/MZero-Labs/ttg/pull/235 Commit in main: https://github.com/MZero-Labs/ttg/commit/4374a3c84029e7a33dcd7c32fc3b1277351a290c	Michael De Luca	Resolved	
I01 #32 Missing override keyword for interface inherited methods	None		While I had not considered the "best practices" angle to this, I question whether it is actually worth doing. The compiler is already checking that all the interface functions are being implemented, but it's just not checking that all the implemented	Michael De Luca	Won't fix	

			functions are in the interface. To do this properly, every single external or public function would need the override keyword, which seems just wasteful and tedious.			
I02 #31 Some contracts don't implement their entire interface	None None	Some abstract contracts don't implement all functions of the interfaces they inherit from	PRs: https://github.com/MZero-Labs/ttg/pull/212 https://github.com/MZero-Labs/common/pull/13 Commits in main: https://github.com/MZero-Labs/ttg/commit/7ae0f718263c22cad52d9c17de1b4eb9cb8a84dd https://github.com/MZero-Labs/common/commit/dbc463849814b71c37f709e663b84e5270a4b8e6 Abstract contracts do not need to implement virtual empty functions overriding the interface, since this is counterproductive and redundant. This has been undone in a subsequent PR.	Pierrick Turelier	Won't fix	
I03 No need to set isActive to false if that mapping entry was deleted #30	None None	Unnecessary resetting to default value	PR: https://github.com/MZero-Labs/protocol/pull/130 Commit in main: https://github.com/MZero-Labs/protocol/commit/8024584e9d3ee0292410ffb00df7927f52acc257	Pierrick Turelier	Resolved	
I04 Unnecessary Access to Storage in MToken._startEarning #26	None None		PR: https://github.com/MZero-Labs/protocol/pull/130 Commit in main: https://github.com/MZero-Labs/protocol/commit/fc9069a2c3b4e2f3de43d4c1e51d3406cc7b21e5	Pierrick Turelier	Resolved	

I05 Unnecessary check in StandardGovernor.state #24	None None		PR: https://github.com/MZero-Labs/ttg/pull/221 Commit in main: https://github.com/MZero-Labs/ttg/commit/768c250da47c2a29a7dfcec3c3d30b5d56ad1294	Pierrick Turelier	Resolved	
I06 #19 Code should not panic underflow	None None	Proper code validation is missing by, instead, relying on panic errors.	PR: https://github.com/MZero-Labs/protocol/pull/138 Commit: https://github.com/MZero-Labs/protocol/commit/20b3b62f7d5dd97f5541f79cef51f8146a2dcbd5 PR: https://github.com/MZero-Labs/common/pull/16 Commit: https://github.com/MZero-Labs/common/commit/160d1058eab98ddb1e0406ae519c13f8b3d9674d PR: https://github.com/MZero-Labs/ttg/pull/225 Commit: https://github.com/MZero-Labs/ttg/commit/e147ec4aa99b7b95482ab4d193cd44ab41143d9f	Antonina Norair	Resolved	
I07 #18 StartedEarning event is emitted even if account is already earning	None None	The *StartedEarning* and *StoppedEarning* events are always emitted when calling *startEarning* and *stopEarning* respectively, even if the caller is already in this state	PR: https://github.com/MZero-Labs/protocol/pull/125 Commit in main: https://github.com/MZero-Labs/protocol/commit/5b68e4e70a09b13e49b82267442b30555a083ddf	Pierrick Turelier	Resolved	

I08 #17 Wrong comment in Standard Governor's execute function	None None		PR: https://github.com/threesigmaxyz/mzero-labs-8-1-2024-issues-external/issues/17 Commit in main: https://github.com/MZero-Labs/ttg/commit/22bd53c50b4224217a4548d85c9ac8615f49f31f	Michael De Luca	Resolved	
I09 #15 Unnecessary currentEpoch zero check in StandardGovernor and ThresholdGovernor	None None		PR: https://github.com/MZero-Labs/ttg/pull/221 Commit in main: https://github.com/MZero-Labs/ttg/commit/1c7576ff2a32fe37b095c79d1d5262beb92b6f52	Michael De Luca Pierrick Turelier	Resolved	
I10 #12 Inconsistent naming of function in PureEpochs	None None		PR: https://github.com/MZero-Labs/ttg/pull/221 Commit in main: https://github.com/MZero-Labs/ttg/commit/63b247c71c2d53ad3d648d58b5dd02deed7ac1a4	Michael De Luca Pierrick Turelier	Resolved	
I11 _checkAndIncrementNonce in ERC5805 raises a ReusedNonce error for non used nonces	None None	ReusedNonce custom error is misleading since the error could also be raised if the nonce is in the future and not current	PRs: https://github.com/MZero-Labs/common/pull/13 https://github.com/MZero-Labs/ttg/pull/212 Commits in main: https://github.com/MZero-Labs/common/commit/d75ef4aff5b7d9a06ed13ca8dc89c9ecc470baad https://github.com/MZero-Labs/ttg/commit/4e20a80138557eb27fc0650d11f7a4b1a0916bfe	Pierrick Turelier	Resolved	

I12 #9 castVoteWithReason always fires a VoteCast event with an empty reason	None None	Empty reason	PR: https://github.com/MZero-Labs/ttg/pull/231 Commit in main: https://github.com/MZero-Labs/ttg/commit/47cd714bd503ed10afb4be002ef46318c6cdbc31	Pierrick Turelier	Resolved	Duplicate of ChainSecurity I-09
I13 transferFrom in ERC20Extended will always emit an Approval event if the allowance changes	None None	The *Approval* event should only be emitted when calling *approve()* and not when transferring and increasing the allowance.	PR: https://github.com/MZero-Labs/common/pull/14 https://github.com/MZero-Labs/common/pull/18 Commit in main: https://github.com/MZero-Labs/common/commit/3076d87e1df18eacd5a0c41ccc90367db50ea6f2	Pierrick Turelier	Resolved	
I14 EIP712's _revertIfError should use all SignatureChecker.Error errors	None None	Some types of errors are not explicitly raised in the ERC712 implementation	PR: https://github.com/MZero-Labs/common/pull/13 Commit in main: https://github.com/MZero-Labs/common/commit/7eef72d526cd44995f93fe09f3068ae9a155250f	Pierrick Turelier	Resolved	
I15 The IERC3009 interface is not fully conforming to the standard	None None	Typehash public functions are not implemented in the IERC3009 interface	PR: https://github.com/MZero-Labs/common/pull/13 Commit in main: https://github.com/MZero-Labs/common/commit/ab7366e1616f397d4ac2f6abda7537e42894d8be	Pierrick Turelier	Resolved	

Quantstamp [24 total, report]

Issue	Severity [Declared, Suggested]	Description & Notes	Fix PR / commit	Owner	Status	Duplicate of
H01 MZ-1 Returning Excessively High Unrealized Inflation in Exceptional Scenario	High Low		PR: https://github.com/MZero-Labs/ttg/pull/233/files#diff-767e30400d380a58c1330b6a340ea2b13ffb8544ace23c95f0335e6fde3fbfbdR266-R286 Commit in main: https://github.com/MZero-Labs/ttg/commit/0c86903456266e3814d5086c63b680063d03bdb6	Michael De Luca Antonina Norair	Resolved	Three Sigma L05 ChainSecurity CS-MZEROCORE-014 Independent auditor L03
M01 MZ-2 Risk of Overflow when Minting M Tokens	Medium Low	Example 2.Calling *startEarning* would revert when safe casting to *uint112* in *_getPrincipalAmountRoundedDown()*	PR: https://github.com/MZero-Labs/protocol/pull/126 Commit in main: https://github.com/MZero-Labs/protocol/commit/74523a8e77be8654b902baaca41201d135c72190 This has been re-solutioned in: https://github.com/MZero-Labs/protocol/pull/143/files#diff-6d16a12288164b2eb7971f1325b337cb7ab8909b0fb939f78524943ee2f93d2bR205-R226	Pierrick Turelier	Resolved	Three Sigma H01 Three Sigma L04
M02 MZ-3 Truncation Risk in Arbitrary Integer Casting	Medium	1. *_getInflation()* is called in 2 places, the value passed to it is either already casted to *uint240* or capped to *type(uint240).max* before being passed to the function.	2. PR: https://github.com/MZero-Labs/protocol/pull/128 Commit in main: https://github.com/MZero-Labs/protocol/commit/fdb93ed64fd40294cac9935574d6a038bfacc289	Pierrick Turelier Michael De Luca	1. Won't fix 2. Resolved	

M03 MZ-4 Privileged Roles and Ownership	Medium Informational	Intended behavior		Gregory Di Prisco	Won't fix	
L01 MZ-5 Overflow Risks in Unchecked Arithmetic Operations	Low	1. In the function MinterGateway.maxAllowedActiveOwedMOf(), the multiplication of collateralOf(minter_) by mintRatio() is unchecked. While mintRatio() is capped, the product can theoretically exceed the maximum uint256 value, especially when collateralOf returns a high uint240 value. 2. In the function EpochBasedInflationaryVoteToken._getUnrealizedInflation(), the addition of inflation_ to the balance_ and the subsequent inflation calculation is unchecked. This can lead to overflow, particularly when the values approach the maximum of uint240. 3. In the function ContinuousIndexingMath.divideDown(), the multiplication with EXP_SCALED_ONE can overflow. 4. In the function ContinuousIndexingMath.divideUp(), the multiplication with EXP_SCALED_ONE can overflow. 5. In the function ContinuousIndexingMath.multiplyIndices(), the multiplication can overflow. 6. In the function PowerToken._getVotes(), the addition of the bootstrapped balance and the unrealized inflation can overflow as the values approach the maximum of uint240.	PR: https://github.com/MZero-Labs/ttg/pull/233/files#diff-767e30400d380a58c1330b6a340ea2b13ffb8544ace23c95f0335e6fde3fbfbdR266-R286 Commit in main: https://github.com/MZero-Labs/ttg/commit/0c869034562266e3814d5086c63b680063d03bdb6	Michael De Luca Antonina Norair	Resolved	
L02 MZ-6 Precision Loss in Token Distribution Affects Small Holders	Low	Dust/lost token is inevitable in any system since divisions are unavoidable. Dust could be lost by sending to the wrong/incompatible address as well. The best solution to somewhat mitigate dust is to accumulate distributions with scaling over various epochs, and then divide by the scale. Effectively, the more epochs one claims over, the more "dust" will be added together to form whole units that can be claimed. See the PR.	PR: https://github.com/MZero-Labs/ttg/pull/234 Commit https://github.com/MZero-Labs/ttg/commit/3ae0bd3eda9d0399074549bdd9cdfe5e2f064ebf	Michael De Luca	Resolved	Three Sigma L03

L03 MZ-7 Lack of Clear Error Messages in _execute Function	Low Informational	Parent function will revert with `ProposalCannotBeExecuted` Clarity was improved by the following pr	PR: https://github.com/MZero-Labs/ttg/pull/203 Commit in main: https://github.com/MZero-Labs/ttg/commit/f39adbff195cfd20e9880ba071d0f7e81284458a	Michael De Luca	Resolved	
L04 MZ-8 Power Token Supply Can Be Bypassed Through Re-Entrancy	Low	Not a usecase of reentrancy, token supply won't be be incorrectly increased. Also cash token will be limited to pre-defined set of non-malicious tokens Nonetheless I will flip the order of mint and transferFrom	PR: https://github.com/MZero-Labs/ttg/pull/223 Commit in main: https://github.com/MZero-Labs/ttg/commit/28913e3f94217827fb4f568791e8ceb5226052d9	Antonina Norair	Resolved	
L05 MZ-9 Ambiguous Error Messaging in MinterGateway.proposeMint()	Low Informational	Undercollateralized error has a field for `maxAllowedActiveOwedM`. If it is 0, and activeOwedM > 0, it means minter didn't update their collateral on time.		Antonina Norair	Acknowledged, won't fix	
L06 MZ-10 - Signatures Missing Expiry	Low Low	No *expiry* parameter is passed to *castVoteBySig()* and thus it is not possible for the signer to precise when the signature should be deemed invalid. I don't think we should fix it since the signature will be invalid once the vote has passed and this functionality is not present in OZ Governor.		Pierrick Turelier	Acknowledged, won't fix. We are following the OZ Governor standard: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/bd325d56b4c62c9c5c1aff048c37c6bb18ac0290/contracts/governance/IGovernor.sol#L291	
L07 MZ-11 Functions for Getting Vote Token Delegatees and Vote Power Amounts May Revert if Gas Too High		The functions of EpochBasedVoteToken : 1. pastBalanceOf(), 2. pastDelegates(), 3. getPastVotes(), and 4. pastTotalSupply(), Use a binary search instead in order to arrive at the entry for the correct epoch.		Antonina Norair Pierrick Turelier Michael De Luca	Won't fix	Duplicate of OZ M-05

L08 MZ-12 - Missing Input Validation	Low Low	Missing input validations in constructor 1.1. *voteToken* is checked in BatchGovernor 3.1. & 2. Signature length is checked in the calling functions *recoverECDSASigner* and *isValidECDSASignature* 4.1. & 2. The zero address won't be able to spend the tokens, so it doesn't make sense to check for *address(0)* 5.1. Not really an issue and we save gas by not performing a sload.	PRs: https://github.com/MZero-Labs/common/pull/25 https://github.com/MZero-Labs/protocol/pull/147 https://github.com/MZero-Labs/ttg/pull/237 Commits in main: https://github.com/MZero-Labs/protocol/commit/a7925aa8cd5b4b36fd9f5c35cb947325d206f325 https://github.com/MZero-Labs/common/commit/0da7d2fecff44dc6d9d4c3d9b3cf6b8dd8926f9d https://github.com/MZero-Labs/ttg/commit/0c869034562666e3814d5086c63b680063d03bdb6	Pierrick Turelier	Resolved	5.2 Duplicate of L07 Open Zeppelin
I01 MZ-13 Rounding Mismatch in M Token Minting and Debt Calculation	Informational	The MinterGateway contract introduces a rounding discrepancy between the calculation of M tokens owed and the number of M tokens minted. Specifically, while the protocol rounds up the principal amount to determine the M tokens owed, it rounds down during the minting process. This can result in minters accruing debt for M tokens that are not minted.	We acknowledge 1 unit discrepancy that can occur for the minter. It is the result of rounding up, down in favor of protocol and storing owed M in form of principal	Antonina Norair	Won't fix, by design	
I02 MZ-14 Risks in Validator-Based Collateral Attestation with Off-Chain Dependencies	Informational	File(s) affected: MinterGateway.sol Description: The protocol's method for updating on-chain Collateral Value, crucial for M token generation, is contingent on validators' attestation			Won't fix, by design	

		of off-chain data. This reliance introduces potential risks due to dependencies on the accuracy of off-chain data and external validation systems. Moreover, the lack of on-chain incentives for validators, who operate based on off-chain agreements, may influence the reliability of the validation process.				
I03 MZ-15 Risk of Standard Governance Griefing Through Excessive Low-Cost Proposals	Informational	Description: The standard governance permits any user to submit proposals with a refundable fee. The Zero Governance control permits this fee to be set to zero or a very low value. While this inclusivity is beneficial for broad participation, it opens the door to governance griefing. Malicious actors can exploit the low barrier to entry by flooding the governance with spam proposals. This can overwhelm token holders, who must review and respond to each proposal to avoid dilution of their power tokens.			Won't fix, by design	
I04 MZ-16 Limitation of Padé Approximant in Approximating Exponential Functions for Financial Calculations	Informational	The use of the Padé approximant to approximate the exponential function in compound interest calculations may lead to accuracy limitations. The approximation starts to show a noticeable deviation of approximately 1 basis point from the actual value of .	Padé was chosen because it is accurate enough while being cheaper, but more importantly, never exploding/overflowing/reverting for all inputs, since the resulting value plateaus (before dropping back down).	Michael De Luca	Won't fix	
I05 MZ-17 Gas Optimization: Cache Variables	Informational	Repeatedly accessing certain variables can be gas-intensive. To optimize gas usage, values frequently accessed should be stored in memory variables. Consider storing the value in a memory variable and reference the	1, 2 and 3. For loops have been optimized in recent versions of Solidity and there is no more saving in gas when caching array.length. 4. It costs 7 more gas when computing the *currentIndex* if we	Pierrick Turelier	Won't fix	

		memory variable for the following variables: 1. proposalIds.length from BatchGovernor._castVotes() 2. proposalIds.length from StandardGovernor._castVotes() 3. allowedCashTokens_.length from ZeroGovernor.constructor() 4. uint256(x) from ContinuousIndexingMath.exponent()	cache *x* in a variable.			
I06 MZ-18 Loss of Precision Due to Division before Multiplication	Informational	Division before multiplication may result in a loss of precision when the operations are carried over integer numbers. This occurs at StableEarnerRateModel.sol#L106: int256 lnArg_ = int256(1e12 + ((((uint256(totalActiveOwedM_) * (deltaMinterIndex_ - 1e12)) / 1e12) * 1e12) / totalEarningSupply_));	PR: https://github.com/MZero-Labs/protocol/pull/124 Commit in main: https://github.com/MZero-Labs/protocol/commit/cac3c24b038d9d6c71bad1132b9f4617e91ef2f1	Antonina Norair Michael De Luca	Resolved	Duplicate of https://github.com/threesigmaxyz/mzero-labs-8-1-2024-issues-external/issues/33
I07 MZ-19 Clone-and-Own	Informational	The clone-and-own approach involves copying and adjusting open source code at one's own discretion. The following instances have been identified: 1. SignatureChecker.sol: OpenZeppelin ECDSA.sol and SignatureChecker.sol. 2. UIntMath.sol: OpenZeppelin SafeCast.sol. 3. ERC712.sol: OpenZeppelin EIP712.sol. 4. ERC20Extended.sol: OpenZeppelin ERC20.sol and ERC20Permit.sol.	We made the deliberate approach to create our own contracts that we can extend with the latest EIPs instead of relying on third party libraries. Also, it allows us to have uniform and detailed error messages, instead of an uncontrolled mix of requires, boolean returns, and reverts.	Pierrick Turelier	Won't fix	
I08 MZ-20 Application Monitoring Can Be Improved by Emitting More Events	Informational	In order to validate the proper deployment and initialization of the contracts, it is a good practice to emit events. Also, any important	disagree strongly with any suggestion to emit redundant events in MToken, because: • they are a waste of gas • they have the potential to	Michael De Luca	Won't fix	

		state transitions can be logged, which is beneficial for monitoring the contract, and also tracking eventual bugs or hacks. Below we present a non-exhaustive list of events that could be emitted to improve application management: 1. MToken._addEarningAmount() : _balances[].rawBalance and principalOfTotalEarningSupply. 2. MToken._addNonEarningAmount() : _balances[].rawBalance and totalNonEarningSupply. 3. MToken._startEarning() : _balances[].rawBalance , principalOfTotalEarningSupply and totalNonEarningSupply. 4. MToken._stopEarning() : _balances[].rawBalance , principalOfTotalEarningSupply and totalNonEarningSupply. 5. MToken._subtractEarningAmount() : _balances[].rawBalance , principalOfTotalEarningSupply. 6. MToken._subtractNonEarningAmount() : _balances[].rawBalance , totalNonEarningSupply. 7. MToken._transferAmountInKind() : _balances[].rawBalance. 8. MinterGateway._imposePenaltyIfMissedCollateralUpdates() : _minterStates[].penalizedUntilTimestamp. 9. MinterGateway._updateCollateral() : _minterStates[].collateral and _minterStates[].updateTimestamp.	• emit intermediate values that should never be emitted • events should emit deltas, not final states, so, for example, how much principal was increased would still require knowledge of the previous amount of principal, which itself is based on a delta and the amount before that, etc, etc. Long story short, anyone that cares to properly index events to track state at any given moment already needs to be aware of how the contract works			
I09 MZ-21 Code Documentation	Informational	We recognized a few places where the code documentation can be improved:	PRs: https://github.com/MZero-Labs/common/pull/17 https://github.com/MZero-Labs/protocol/pull/130	Pierrick Turelier	Resolved	

		1. TTGRegistrarReader.sol#L24: The comment should say earners ignore list rather than earners list. 2. MinterGateway.sol#L98-99: NatSpec comments similar, consider removing one. 3. ThresholdGovernor.sol#L16: Link is outdated and should rather be https://portal.thirdweb.com/contracts/build/base-contracts/erc-20/vote. The following typos were also spotted: 1. ContractHelper.sol#L6 : aan -> an. 2. ContinuousIndexingMath.sol#L88 : costs -> cost. 3. MinterGateway.sol#L1032 : BY -> By.4. EpochBasedVoteToken.sol#L285 : array of given by -> by. 5. EpochBasedInflationaryVoteToken.sol#L12 : ,a nd -> , and. 6. EpochBasedInflationaryVoteToken.sol#L116 : the its -> its. 7. PowerToken.sol#L309 and L312 : that in -> that is.	https://github.com/MZero-Labs/ttg/pull/221 Commits in main: https://github.com/MZero-Labs/protocol/commit/5445cff372e13798b31969ed5a35fcfb40fea9ea https://github.com/MZero-Labs/ttg/commit/829db205a5b716ec80059aa5451a03a4c6422ce5 https://github.com/MZero-Labs/common/commit/0a0cae40c2c88625cb455fd41bb2a5740f85a7d3			
I10 MZ-22 Undocumented Magic Constants	Informational	To improve readability and lower the risk of introducing errors when making code changes, it is advised to not use magic constants throughout code, but instead declare them once (as constant and commented) and use these constant variables instead. The following instances should therefore be changed accordingly: 1. StableEarnerRateModel.sol#L106 : 1e12.	1 - 1e12 will be removed. PR: https://github.com/MZero-Labs/protocol/pull/130 Commit in main: https://github.com/MZero-Labs/protocol/commit/fa751d7bf476d738a6b010a46da8afe772b16b5b	Pierrick Turelier	Resolved	

		2. StableEarnerRateModel.sol#L109 : 1e6.				
I11 MZ-23 Outstanding Todo Comments	Informational	Pending TODOs in code	PR: https://github.com/MZero-Labs/ttg/pull/222 Commit in main: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c	Antonina Norair	Resolved	OZ I01
I12 MZ-24 Adherence to Specification	Informational	Description: We identified a number of occurrences where the code does not match the specification provided.	1, 5 and 6 - These functions will be reworked. 2. Is this mentioned in the whitepaper? I don't see it in the code. 7. Only applies for MinterGateway. PR: https://github.com/MZero-Labs/protocol/pull/130 Commit in main: https://github.com/MZero-Labs/protocol/commit/5828c44739fac3f7799171072410aad1ca6c8d96	Pierrick Turelier Antonina Norair	Resolved Antonina Norair re-review and adjust engineering specs before Sherlock coding competition	

Certora [13 total, report]

Issue	Severity [Declared, Suggested]	Description & Notes	Fix PR / commit	Owner	Status	Duplicate of

C01 Attacker can double it's PowerToken balance and voting power every time Reset event occurs	Critical High	Double bootstrap was happening in self-delegation or self-transfer Delegator == delegatee Sender == recipient	PR: https://github.com/MZero-Labs/ttg/pull/215 Commit in main: https://github.com/MZero-Labs/ttg/commit/c16400216827b6a6a5485823c2c283c35cb49e75	Antonina Norair	Resolved	
C02 Attacker can inflate his PowerToken balance by repeatedly claiming historic inflation with sync() function	Critical High, introduced by fix		Commit in main: https://github.com/MZero-Labs/ttg/commit/6b487a018a5b4496484b76a0815ee8a9577214d2	Michael De Luca	Resolved	
H01 Past Voting Power isn't read from bootstrap after sync	High		Commit in main: https://github.com/MZero-Labs/ttg/commit/6b487a018a5b4496484b76a0815ee8a9577214d2	Michael De Luca Antonina Norair	Resolved	
M-01: A changed rate would not take effect until the updateIndex() function is called	Medium Info		Design consideration. Governance module and TTG are decoupled. Protocol is read only for TTG.		Won't fix	
M-02. One rogue validator might allow a minter to avoid paying penalties	Medium Info		Resolved, but also not really possible by offchain design considerations: PR: https://github.com/MZero-Labs/protocol/pull/114 Commit in main: https://github.com/MZero-Labs/protocol/commit/b2c421c132cf6af6a18860ad17285b900be83163		Resolved	
L01 Overflow check reverts due to overflow without emitting a revert message	Low	Missing uint256 cast before addition	PR: https://github.com/MZero-Labs/protocol/pull/126 Commit in main:	Antonina Norair Pierrick Turelier	Resolved	

			https://github.com/MZero-Labs/protocol/commit/bdd94b9952952c525f9e91ccbd762af792da7d4b			
L02 Proposal fee is not returned to proposer after RESET event	Low	In the current implementation proposal will Expire and fee will be sent to the distribution vault.		Michael De Luca Pierrick Turelier Antonina Norair	Won't fix, design	
L03 _lastSyncs array is redundant, resulting in waste of gas with any access or write operation	Low		Commit in main: https://github.com/MZero-Labs/ttg/commit/6b487a018a5b4496484b76a0815ee8a9577214d2	Michael De Luca	Resolved	
I01 Lack of EIP-712 compliance: using keccak256() directly on an array or struct variable	Informational		PRs: https://github.com/MZero-Labs/protocol/pull/120 https://github.com/MZero-Labs/ttg/pull/208 Commits in main: https://github.com/MZero-Labs/protocol/commit/287efbd7d4c8ba0fbbc14f1dae71c494d6875b70 https://github.com/MZero-Labs/ttg/commit/5d6ba760fc764617f3997e7db060ea06e79ff476		Resolved	Duplicate of ChainSecurity M02
I02 Array lengths not checked	Informational		PR: https://github.com/MZero-Labs/ttg/pull/222 Commit in main: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c		Resolved	
I03 Certain functions should not be marked as payable	Informational	Fully compatible with OZ Governor interfaces			Won't fix	

I04 Unnecessarily small uint type for an incremented variable	Informational		Type is sufficient enough for nonces.		Won't fix	
I05 Possible gas griefing when emitting voting event	Informational	Arbitrarily long reason string which could make emitting of an event very gas costly for relayer.	M^0 doesn't plan to provide relaying of castVotes. Possible relayers should take those risks into consideration.		Won't fix	

Chainsecurity [23 total, report]

Issue	Severity [Declared, Suggested]	Description & Notes	Fix PR / commit	Owner	Status	Duplicate of
M01 Code Restricts Execution of Proposal to 1Epoch	Medium Info, inconsistency between papers	The function StandardGovernor.execute() tries to execute all proposals voted in the last two epochs. However, the function StandardGovernor.state() returns the status Succeeded only for proposals voted in the previous epoch. The state Expired is returned for older proposals, hence stopping them from being executed. This behavior conflicts the whitepaper and the code comments in function execute() which state that a successful proposal can be executed during the next 2 epochs: // Proposals have voteStart=N and voteEnd=N, and can be executed only during epochs N+1 and N+2.	PR: https://github.com/MZero-Labs/ttg/pull/206 Commit in main: https://github.com/MZero-Labs/ttg/commit/22bd53c50b4224217a4548d85c9ac8615f49f31f	Michael De Luca	Resolved	

M02 EIP-712 Dynamic Types	Medium Medium?	The EIP-712 is not fully compliant with the standard. It must encode dynamic types	PRs: https://github.com/MZero-Labs/protocol/pull/120 https://github.com/MZero-Labs/ttg/pull/208 Commits in main: https://github.com/MZero-Labs/protocol/commit/287efbd7d4c8ba0fbbc14f1dae71c494d6875b70 https://github.com/MZero-Labs/ttg/commit/5d6ba760fc764617f3997e7db060ea06e79ff476	Pierrick Turelier	Resolved	
M03 Earner Interest Can Exceed Minter's Interest	Medium Low	Using SplitEarnerRateModel can lead to overprinting of M		Antonina Norair	Won't fix, use StableEarnerRateModel	
M04 Effects of Roundings in PowerToken	Medium	Michael De Luca seems similar to prototech reportings Explanation is needed		Michael De Luca	Won't fix Initial supply simulations are needed (March)	
M05 Standard Proposal Fee Has Ambiguous Denomination	Medium Info?	The Standard Proposal Fee can be changed in two ways: ZeroGovernance: setCashToken(newCashToken_, newProposalFee_) Standard or Emergency governances: setProposalFee() The second option changes the proposal fee and keeps the current cash token in place.	As mentioned above switch of cash token is coupled with switch of proposal fee value to avoid unfair griefing attacks	Michael De Luca	Won't fix, design	
L01 Contract ERC3009 Inherits StatefulERC712	Low	The contract ERC3009 extends the abstract contract StatefulERC712 which keeps track of used nonces in the public mapping nonces. However, ERC3009 does not use any functionality of this contract. Furthermore, ERC3009 uses	ERC20Extended inherits from ERC3009. Indeed, ERC3009 does not need StatefulERC712 but ERC20Extended does. For this reason, we won't proceed to this change.	Pierrick Turelier	Won't fix	

		random nonces of type bytes32 and the standard explicitly avoids sequential nonces. On contrary, StatefulERC712 is designed to use sequential nonces. Hence, extending ERC712 is enough.				
L02 EIP5805 DelegateChanged Not Always Emitted	Low	The EIP-5805 specs requests the DelegateChanged event to be emitted when delegator changes the delegation of its assets from fromDelegate to toDelegate. The function EpochBasedVoteToken._setDelegatee does not fully adhere to the standard as only emits the event when it is not the first change of delegation. I.e., if _delegatees[delegator_].length==0 the function starts a snapshot for the account with the new delegatee and returns without emitting the event.	PR: https://github.com/MZero-Labs/ttg/pull/220 Commit in main: https://github.com/MZero-Labs/ttg/commit/27c751f6177850751053c011b3a0327896db3e44	Pierrick Turelier	Resolved	
L03 Excess Owed M Can Be Larger Due to Rounding	Low	Function MToken.totalSupply() rounds down the total supply of earners, therefore the excess M amount is computed slightly larger than the real value. In this case, the gateway will mint more tokens to the vault.	We use `getPresentAmountRoundedDown` to offset this situation while calculating `totalOwedM`	Michael De Luca Antonina Norair	Won't fix	
L04 Inactive Minters Can Be Frozen	Low	The function MinterGateway.freezeMinter allows approved validators to freeze arbitrary addresses. But nothing prevents a non-active minter to be frozen.	This is by design. It allows for example a Validator to freeze a Minter that is not active yet, while the Minter is waiting to find a Validator to be able to update his collateral. In the meantime, the Minter can activate his account and not be penalized for missing to update his collateral.	Pierrick Turelier	Won't fix	

L05 Incomplete Interfaces	Low	1. The contract MinterGateway is ContinuousIndexing and ERC712, but IMinterGateway only extends IContinuousIndexing. For completeness, IMinterGateway should also inherit IERC712. 2.The interface IERC3009 should declare the functions TRANSFER_WITH_AUTHORIZATION_TYPEHASH() and RECEIVE_WITH_AUTHORIZATION_TYPEHASH() to match the ERC-3009 interface standard.	1. PR: https://github.com/MZero-Labs/protocol/pull/130 Commit in main: https://github.com/MZero-Labs/protocol/commit/ccf24b640e89f3e53035cbc02c1bd8c093cd134b	Pierrick Turelier	Resolved	2. Duplicate of ThreeSigma I15
L06 Inconsistent Collateral and Penalty at Expiry Boundary	Low	The penalty and collateral calculation are not consistent with each other when block.timestamp == updateTimestamp + updateCollateralInverval. The collateral will still be the non-zeroed collateral value, but a penalty for missed collateral update will still be charged for 1 period.	PR: https://github.com/MZero-Labs/protocol/pull/137 Commit in main: https://github.com/MZero-Labs/protocol/commit/79303b23bb2422dd2fb6181512ed90d9e022c2ee	Antonina Norair	Resolved	
L07 Incorrect Specifications	Low	1. The natspec description of IRateModel.rate states that the return value is APY in BPS. However, rate() returns the yearly interest rate does not consider the compounding. 2. The natspec description for principalAmount in event PenaltyImposed is incorrect. 3.The natspec @return weight_ of BatchGovernor._castVote()	PRs: https://github.com/MZero-Labs/protocol/pull/130 https://github.com/MZero-Labs/ttg/pull/221 Commits in main: https://github.com/MZero-Labs/protocol/commit/0882c67911af94e3180296166b9d3bda9ca7ea24 https://github.com/MZero-Labs/ttg/commit/8281e309b265761be903595b0ceb7b21bba30420	Pierrick Turelier	Resolved	

		indicates The type of support to cast for each proposal, but it should be the voting power of the voter.				
L08 Missing Input Sanitization		- Some of the functions accept an epoch=0 as input, which is an invalid input as *epoch>0.*	PR: https://github.com/MZero-Labs/ttg/pull/221 Commit in main: https://github.com/MZero-Labs/ttg/commit/b5a7eb88a56aaeab9741a3f99468b17ebab44058	Pierrick Turelier	Resolved	
L09 No Expiry in Buy Function	Low	The function PowerToken.buy() does not allow users to specify an expiry timestamp, which would prevent a transaction to execute at a later time. Currently, it is possible that user's transaction gets executed at a future transfer epoch and potentially buys tokens with a price higher than originally intended.	PR: https://github.com/MZero-Labs/ttg/pull/216 Commit in main: https://github.com/MZero-Labs/ttg/commit/3fb74e72f03d45e7ec56f5c420143bcb627ac206	Antonina Norair	Resolved	
L10 Possible Overflow When Syncing Accounts (bug in unrealizedInflation calculations)	Low	The function _sync() in EpochBasedInflationaryVoteToken computes the unrealized inflation of an account by iterating through all epochs since last sync. The for-loop is implemented in _getUnrealizedInflation() and in each iteration, except the last one, it checks that the new balance does not exceed the limits: However, if the inflation from the last iteration causes the final balance of an account to exceed the limit (type(uint240).max), function _sync() updates the balance with the full inflation amount via _addBalance(). The later uses unchecked block, hence an overflow happens.	PR: https://github.com/MZero-Labs/ttg/pull/233/files#diff-767e30400d380a58c1330b6a340ea2b13ffb8544ace23c95f0335e6fde3fbfbdR266-R286 Commit in main: https://github.com/MZero-Labs/ttg/commit/0c86903456266e3814d5086c63b680063d03bdb6	Michael De Luca Antonina Norair	Resolved	Duplicate of ThreeSigma L05 Quantstamp H01 ChainSecurity CS-MZEROCORE-014 Independent auditor L03

L11 Possible Overflow in convertToBasisPoints	Low	The function ContinuousIndexingMath.convertToBasisPoints() uses unchecked block to convert a uint64 input into a uint32 type. The computation can overflow for large values of input, i.e., input > type(uint32).max * 10**8. This issue is unlikely to happen in the current codebase as the function is called only with inputs representing interest rates which are capped.	PR: https://github.com/MZero-Labs/protocol/pull/143 Commit in main: https://github.com/MZero-Labs/protocol/commit/cf5463f665aeeaf4278cb783ca41791001621cff	Michael De Luca Antonina Norair SAME AS L18, SYNC NEEDED Michael De Luca let's review uint types once again	Resolved	Duplicate of L18
L12 Possible Rounding to 0 in getSafeEarnerRate	Low	The function getSafeEarnerRate in StableEarnerRateModel computes the value lnArg_ as follows: Note that deltaMinterIndex_ is usually close to 1 (10**12) for short time intervals, hence deltaMinterIndex_ - 1e12 is a value close to 0. Therefore the intermediary result (uint256(totalActiveOwedM_) * (deltaMinterIndex_ - 1e12)) / 1e12 rounds down to 0 for values of totalActiveOwedM below a certain threshold.	Precision improved with PR: https://github.com/MZero-Labs/protocol/pull/124 Commit in main: https://github.com/MZero-Labs/protocol/commit/cac3c24b038d9d6c71bad1132b9f4617e91ef2f1	Michael De Luca Antonina Norair	Won't fix, considering different model	Duplicate of ThreeSigma L01
L13 Reentrancy in PowerToken Re-Buy	Low	In the function PowerToken.buy() the cashToken in transferred from the buyer before the totalSupply of the token is increased by mint(). If the cashToken implements callbacks (ERC777-like), this enables a reentrancy issue that allows an attacker to mint arbitrary amounts of PowerToken, as the amountToAuction would only be decreased after mint() is called.	PR: https://github.com/MZero-Labs/ttg/pull/223 Commit in main: https://github.com/MZero-Labs/ttg/commit/28913e3f94217827fb4f568791e8ceb5226052d9	Antonina Norair	Resolved	Duplicate of Quantstamp L04 (MZ-08)

L14 Remaining Dust in Distribution Vault	Low	Function DistributionVault.getClaimable() rounds down when computing the amount of cash token that can be claimed by an account, hence dust remains in the vault: The dust of cash tokens (including MToken) accumulates in the vault and cannot be withdrawn. In case of MToken, the locked dust has implications for last minters, who might be unable to fully repay their debt and close their positions.	PR: https://github.com/MZero-Labs/ttg/pull/234 Commit in main: https://github.com/MZero-Labs/ttg/commit/3ae0bd3eda9d0399074549bdd9cdfe5e2f064ebf	Michael De Luca	Resolved	Duplicate of ThreeSigma L03
L15 Remaining ToDos in Codebase	Low	Remaining TODO comments	PR: https://github.com/MZero-Labs/ttg/pull/222 Commit in main: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c	Antonina Norair	Resolved	
L16 Timestamp 0 in Signatures	Low	Function MinterGateway.verifyValidatorSignatures() currently allows signatures with a timestamp set to 0. Although it is anticipated that validators will not typically generate signatures with a timestamp of 0, in the event that such signatures occur, there is a risk of replaying them, given that minTimestamp will always be block.timestamp.	PR: https://github.com/MZero-Labs/protocol/pull/114 Commit in main: https://github.com/MZero-Labs/protocol/commit/b2c421c132cf6af6a18860ad17285b900be83163	Michael De Luca Antonina Norair	Resolved	Duplicate of Three Sigma C01 OZ H01
L17 Transfer of Whole Balance Can Revert for Earners	Low	The accounting of MToken balance for accounts in the earning state is in principal. When such accounts transfer out MTokens, the principal amount is updated. The function _transfer() always rounds up when the sender is in the earning state, for example:	Check test testFuzz_transfer_wholeBalance PR: https://github.com/MZero-Labs/protocol/pull/148 Commit in main: https://github.com/MZero-Labs/protocol/commit/bdbb59970042855a027f8656578e5dbce853cee0	Michael De Luca Antonina Norair	Won't fix, Didn't reproduce	

		If the account transfers everything, i.e., balanceOf(), the function can revert due to rounding up which might cause an underflow.				
L18 Wrong Condition in StableEarnerRateModel	Low	The function StableEarnerRateModel.getSafeEarnerRate() implements the check expRate_ > type(uint64).max to return early if the rate is too big, otherwise the value returned by ContinuousIndexingMath.convertToBasisPoints(uint64(expRate_)) will be returned. As mentioned in the issue Possible Overflow in ConvertToBasisPoints, the function ContinuousIndexingMath.convertToBasisPoints will overflow if its input is greater than type(uint32).max. The currently implemented check leaves a hole for the values of the rate between type(uint32).max and type(uint64).max where the function will overflow. The function StableEarnerRateModel.getSafeEarnerRate() should return early if expRate_ > type(uint32).max instead.	PR: https://github.com/MZero-Labs/protocol/pull/143 Commit in main: https://github.com/MZero-Labs/protocol/commit/cf5463f665aeeaf4278cb783ca41791001621cff	Antonina Norair Michael De Luca SAME AS L11 , sync needed	Resolved	Duplicate of L11
O01 Consistent Function Naming	Open Q	In the contract EpochBasedVoteToken, the function getPastVotes is named after ERC-5808. For consistency, would it make sense to rename the functions EpochBasedVoteToken.[pastBalanceOf, pastDelegates, pastTotalSupply] to EpochBasedVoteToken.[getPastBalanceOf, getPastDelegates, getPastTotalSupply] ?	I tend to prefer function names without prefix since it is pretty self explanatory that *pastBalanceOf* will return the past balance. Unfortunately we can't rename *getPastVotes* but I don't think we should rename all functions to follow this convention.	Pierrick Turelier	Won't fix	

O02 Edge Case in mintM	Open Q	In the function MinterGateway.mintM, the following check is implemented: The system does not allow inactive owed M to be reactivated and flow back into the principal of total active owed M. Can you elaborate on the reasoning behind this check?	The comment above is a bit misleading. This check is here to ensure that *principalOfTotalOwedM* will not overflow *uint112* since all principal amounts are casted to *uint112*. PR: https://github.com/MZero-Labs/protocol/pull/133 Commit in main: https://github.com/MZero-Labs/protocol/commit/4ccf9b6fcc6ad11d8d86064c971885bef5e32878	Pierrick Turelier	Resolved	
O03 Intended Use of Power and Zero Tokens	Open Q	Governance tokens Zero and Power play two important roles in the system: • Maintain the protocol by voting on proposals. • Claim M token rewards from the vault that help minters close their positions. However, both tokens are implemented as ERC20 tokens and can be deposited in 3rd-party protocols (such as DEXes or lending protocols). If this happens, there are severe consequences for the system, as attacks that overtake governance majority become feasible (e.g., borrowing large amount of tokens in the last block of an epoch). Also, parts of rewards in the distribution vault might get locked. What are your assumptions on the usage of governance tokens outside the system?	We don't expect ZERO or especially POWER tokens being actively used in outside protocols.	Antonina Norair Michael De Luca	Answered	

O04
 	
Larger Pending Retrievals Than Collateral
	Open Q	The function MinterGateway.collateralized() performs the following check:
		
		`// Invariant: total pending retrievals cannot ever be greater than total collateral, else this collateral is zero`
		`if (totalPendingRetrievals_ > totalCollateral_) return 0;`
		
		Can you describe when this scenario can happen and what is the intended behavior?
			This check is very conservative since it would revert with the error *RetrievalsExceedCollateral* in *proposeRetrieval* if trying to retrieve more than the current collateral.
			But this way we ensure that the unchecked subtraction below won't silently underflow.
				Pierrick Turelier	Answered	

O05 Limitations on Propose/Execute Parameters	Open Q	The function propose in both ThresholdGovernor and StandardGovernor takes as input 3 arrays: targets_, values_ and callDatas_. The internal function _propose restricts the values that can be passed in these arrays. For instance targets_ should have only one address and it should be address(this), values_ should have only one 0, while callDatas_ should have only element and start with one whitelisted function selector. Similarly, execute() is payable but the code reverts if msg.value is non-zero. Given the existing limitations, what is the reason for using the existing parameters in function propose() and marking execute() as payable?	We want to stay compatible with the standard OZ Governor and that's why we accept all these parameters in *propose* and why *execute* is payable.	Pierrick Turelier	Answered	
O06 Minter's Wallet Is Continuously Used	Open Q	The function updateCollateral() requires minters to submit transactions to the smart contract on a daily basis. Considering that minter's account is valuable in the system and it should be carefully protected (e.g., as a cold wallet), this might cause inconvenience to minters. Have you considered using a different approach in which minter's wallet does not need to be active continuously?	Ideally, a Minter's wallet should only be used to interact with the MinterGateway. M minted by the Minter, should be sent to another wallet by passing a different destination address in *proposeMint*. When the Minter wants to retrieve their collateral, they will either have to ask the owner of the mint M tokens to call the *burnM* function or acquire M on the secondary market to burn them themselves.	Pierrick Turelier	Answered	

			Of course, Minters should first and foremost ensure that they have established good security practices to avoid any issues.			
O07 Order of Parameters in COUNTING_MODE	Open Q	The function BatchGovernor.COUNTING_MODE() returns the string support=for,against&quorum=for, but the enum VoteType has No in first position and Yes in second position. How does the front-end interprets the returned string? Would it make sense to return support=against,for&quorum=for instead?	PR: https://github.com/MZero-Labs/ttg/pull/246 Commit in main: https://github.com/MZero-Labs/ttg/commit/6f25908f5de5885df392584e0153ed400d4ed1f0	Pierrick Turelier	Resolved	
O08 Recovery When System Incurs Losses	Open Q	It is possible that the system could mint uncollateralized MTokens when more interest is paid to earners than collected from minters. This could happen if updateIndex() is not called frequently. How does the system recover when such losses happen?	We don't expect such situation to happen. No recovery is intended. Even though protocol index will be updated at least once per day (via minter's updateCollateral), StableEarnerRateModel sets confidence interval == 30 days for extra safety. Preventing such situation is a key for functioning system.	Antonina Norair	Answered	
O09 Resubmission of Signatures and Staleness	Open Q	The function _updateCollateral reverts only if the new timestamp is strictly smaller than the current one. But if the exact same batch of signatures is used, the two timestamps will be equal and the check will pass, even though the result is stale. Why not accept the update iff the new timestamp is strictly bigger than the current one?	Suggestion implemented PR: https://github.com/MZero-Labs/protocol/pull/140 Commit in main: https://github.com/MZero-Labs/protocol/commit/41915281a4a66cc24c4793c46db4d14b2a6ba263	Antonina Norair Michael De Luca	Resolved	
O10 Safe totalSupply	Open Q	As MToken.totalSupply() rounds down the earners' contribution, it could be that the totalSupply is slightly undervalued, hence more M tokens get minted to the	It is done by Rounding down vs rounding up `*getPresentAmountRoundedDown*` to offset this situation while calculating `*totalOwedM*`	Antonina Norair	Answered	Duplicate of CS L03 Excess Owed M Can Be Larger Due to Rounding

		distribution vault. Have you considered using a "safe" totalSupply function when computing the rate and the excess owed, where the earners' contribution is rounded up?				
O11 Snapshots Remain in Storage	Open Q	Multiple contracts within the governance module maintain a record of changes for each account in storage. Typically, this information is only extended when new activity happens, and the old data remains uncleared. Have you explored the possibility of optimizing storage usage by clearing outdated information?	Interesting though, we won't add it because of possible complexities.	Antonina Norair Michael De Luca	Answered, won't fix	
O12 Threshold Governors Can Ignore Majority	Open Q	The governors implementing ThresholdGovernor, i.e. ZeroGovernor and EmergencyGovernor, consider a proposal Succeeded if the ratio $\frac{yesVotes}{totalSupply} \geq quorum$ This means that if quorum <50%, the majority is ignored, and a proposal can pass even if it gets more no votes than yes votes. What are the intended values for threshold ratios and have you considered enforcing stricter limits in the smart contracts?	We debated this situation for some time. And decided to keep the current implementation. 1. Thresholds will be significantly higher than 50% 2. If some reason governance lower thresholds below 50%, the described situation is intended	Antonina Norair Michael De Luca	Answered	
I01 Dead Code	Informational	The libraries PureEpochs and ContinuousIndexingMath implement some unused functions. The unused functions will be ignored by the compiler, but unused code can increase the difficulty of understanding the codebase. The functions are: *ContinuousIndexingMath.exponentAssembly*	PRs: https://github.com/MZero-Labs/ttg/pull/222 https://github.com/MZero-Labs/common/pull/20 https://github.com/MZero-Labs/ttg/pull/226 https://github.com/MZero-Labs/pro	Antonina Norair Pierrick Turelier	Resolved	Duplicate of OZ Independent I05

		PureEpochs.getTimeUntilEpochStart *PureEpochs.getTimeUntilEpochEnds* *PureEpochs.getTimeSinceEpochStart* *PureEpochs.getTimeSinceEpochEnd* *SignatureChecker.isValidECDSASignature(address,bytes32,uint8,bytes32,bytes32)* The function EpochBasedVoteToken._subUnchecked is never used.	tocol/pull/139 https://github.com/MZero-Labs/ttg/pull/231 Commits in main: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c https://github.com/MZero-Labs/ttg/commit/2dc057bc09e92105bbbcb67e0ace91b4a6d6bdb3 https://github.com/MZero-Labs/protocol/commit/da0dcf21be15530e0561c9078d0a3f7551e2474b https://github.com/MZero-Labs/common/commit/1212e8a32afda886d9497970c316f677cc0dc9cf https://github.com/MZero-Labs/ttg/commit/ae89c920bee3e9331c7d42b7da324752226bc4e8			
I02 ERC712 Does Not Implement Extension EIP-5267	Informational	The abstract contract ERC712 does not implement the extension EIP-5267 which aims to improve the integration of EIP-712 signatures with third-party tools.	PR: https://github.com/MZero-Labs/common/pull/17 Commit in main: https://github.com/MZero-Labs/common/commit/0a0cae40c2c88625cb455fd41bb2a5740f85a7d3	Pierrick Turelier	Resolved	
I03 Gas optimizations	Informational	17 points in the audit report	5. Won't fix 6. Won't fix. We are currently reverting early and avoiding going through all the ifs. It would cost more gas if we remove this condition. 9. & 10. Won't fix. The code would become more complex. 11. Won't fix. True but this way it is clear that we are performing *_getPrincipalAmountRoundedUp* 12. Won't fix. This project being	Pierrick Turelier	Resolved	

			open source, anyone could extend BatchGovernor and use *quorumRatio*. 14. Won't fix. This check is here in case the contract is being inherited by another one. 15. & 16. Won't fix. Here in case the contract is not deployed by the deployer. 17. Won't fix. Here in case the contract is called by another one that wouldn't check for address zero. PRs: https://github.com/MZero-Labs/protocol/pull/130 https://github.com/MZero-Labs/ttg/pull/221 https://github.com/MZero-Labs/common/pull/17 https://github.com/MZero-Labs/protocol/pull/136 Commits in main: https://github.com/MZero-Labs/protocol/commit/01a6e5e92663831af8b340f28675c0d8c1e10198 https://github.com/MZero-Labs/ttg/commit/fe9c26347b66b004d6abd726d31257b330c78375 https://github.com/MZero-Labs/common/commit/0a0cae40c2c88625cb455fd41bb2a5740f85a7d3			
I04 Inconsistent events	Informational	1. In the function MinterGateway.updateCollateral(), the order of events does not match the order of changes on-chain. For example, the events would be trigger in the following order: update-penalty-penalty, but the order of execution on-chain is penalty-update-penalty. It is in general good practice to emit the events to match the changes on-chain.	1 - The order of events don't really matter, except if there is a change of variable in the event itself. 2 - Not really an issue and this way if the Minter is already active but this function is called in a batch, we avoid reverting the whole batch. 4 & 5 - Will be refactored. 6 - Not really an issue, this way we record that the user tried to	Pierrick Turelier	Resolved	

		2. Anyone can call the function MinterGateway.activateMinter() for an existing active minter and emit the respective event, although no state changes. 3. The event MintCanceled is emitted if calling the function MinterGateway.cancelMint() with mintId_ = 0 although no such proposal can exist. 4. Functions startEarning() and stopEarning() can be called multiple times for an address to emit the respective events. 5. Functions allowEarningOnBehalf() and disallowEarningOnBehalf() can be called multiple times with no state changes. 6. Function PowerToken.buy() can be called at any time with minAmount_ set to 0, so events Buy and Transfer would be emitted even during voting epochs.	purchase with a *mintAmount* of 0. 3. PR: https://github.com/MZero-Labs/protocol/pull/130 Commit in main: https://github.com/MZero-Labs/protocol/commit/1fb6fb3b6997576456f58396c20e834dee041382			
I05 Metadata of PowerToken	Informational	The name and symbol of PowerToken is hardcoded in its constructor: Therefore, name and symbol will be the same for new tokens if redeployed by governance.	There can only be one Power Token at a time.	Pierrick Turelier	Won't fix.	
I06 Misleading Error Name	Informational	The error ReusedNonce emitted in ERC5805._checkAndIncrementNonce() is misleading, as this error will be emitted for nonce that are >*currentNonce* , which haven't	PR: https://github.com/MZero-Labs/ttg/pull/212 Commit in main:	Pierrick Turelier	Resolved	Duplicate of ThreeSigma I11

		been used yet by definition.	https://github.com/MZero-Labs/ttg/commit/4e20a80138557eb27fc0650d11f7a4b1a0916bfe			
I07 Misleading Natspec Description for _divideUp	Informational	The natspec description of PowerToken._divideUp() is misleading as the function actually rounds up the ratio x/y in BPS. Therefore, the function should not be used with arbitrary inputs as the description might suggest:	No clear, just like Pierrick Turelier I didn't understand the question	Antonina Norair	Not clear	
I08 Possible Griefing With Governance Proposals	Informational	ZeroGovernor and EmergencyGovernor do not implement any measure to prevent attackers from proposing a large number of malicious proposals. Although such proposals do not get executed, assuming they do not receive the threshold of yes votes, they might be used to spam the system and make harder for users to find legit proposals.	Intended behavior, zero and emergency governors proposals are optional to vote, requiring thresholds of votes. Possible filtering can be done on a social level. Gas fees on mainnet eventually should be spam prevention against such attacks	Antonina Norair	Answered	
I09 Reason Ignored in BatchGovernor	Informational	The function BatchGovernor.castVoteWithReason() takes as input a string parameter that represents the reason. This parameter is ignored by the function and the event VoteCast is always emitted with an empty string as reason.	PR: https://github.com/MZero-Labs/ttg/pull/231 Commit in main: https://github.com/MZero-Labs/ttg/commit/47cd714bd503ed10afb4be002ef46318c6cdbc31	Pierrick Turelier	Resolved	Duplicate of ThreeSigma I12

Prototech [32 total, report]

Issue	Severity [Declared, Suggested]	Description & Notes	Fix PR / commit	Owner	Status	Duplicate of

C01 Any action that moves delegation to address(0) will cause that user's funds to be locked.	Critical	Delegation to 0x0 leads to inability to re-delgate or transfer tokens	PR: https://github.com/MZero-Labs/ttg/pull/214 Commit in main: https://github.com/MZero-Labs/ttg/commit/27c751f6177850751053c011b3a0327896db3e44	Antonina Norair Michael De Luca	Resolved	
C02 PowerToken Balances can be double counted	Critical	When a user has not synced their past balance and uses `transfer`, `transferFrom`, `delegateBySig` or `transferWithAuthorization` to delegate or transfer to/from themselves, their `pastBalanceOf` is double counted, once as the sender and once as the recipient giving them twice the tokens they should have. These regressions show this occurring and resulting in one user doubling their tokens	PR: https://github.com/MZero-Labs/ttg/pull/215 Commit in main: https://github.com/MZero-Labs/ttg/commit/c16400216827b6a6a5485823c2c283c35cb49e75	Antonina Norair	Resolved	Certora C-01
C03 PowerToken: Delegation and transfer fails when actor.balance > actor.votes	Critical	Identified a scenario where actor balance > actor votes. In this scenario, delegation and transfer above the vote amount but within the available balance fails with an overflow. **Resolution:** This issue happens when actor.balance > delegatee.votes not when actor.balance > actor.votes Duplicate. This is also caused by the same bug as Prototech C01.	PR: https://github.com/MZero-Labs/ttg/pull/214 Commit in main: https://github.com/MZero-Labs/ttg/commit/27c751f6177850751053c011b3a0327896db3e44	Antonina Norair	Resolved	

H01 ERC3009 validAfter and validBefore are incorrectly implemented as inclusive	High	ERC3009 authorizations must not be valid *at* timestamp and must be non inclusive	PR: https://github.com/MZero-Labs/common/pull/13 Commit in main: https://github.com/MZero-Labs/common/commit/ed02a2c94bb22df03b93fe397e73caa2aef5d955	Pierrick Turelier	Resolved	
H02 MToken.mint() can overflow totalNonEarningSupply and principalOfTotalEarningSupply	High	https://github.com/MZero-Labs/protocol/blob/3499f50ff3382729f3e59565b19386ba61ef8e36/src/MToken.sol#L217	Multiple prs Properly resolved here: https://github.com/MZero-Labs/protocol/pull/143/files#diff-6d16a12288164b2eb7971f1325b337cb7ab8909b0fb939f78524943ee2f93d2bR205-R226		Resolved	
H03 PowerToken: Inflation rounding creates deviation in account balances and total supply.	High	Several regressions were discovered that indicate that the total sum of user balances after inflation do not equal the total supply. This is likely due to the necessity of rounding down on the 10% epoch inflation. Example regressions: PASS: test_regression_invariant_P_B1_4d72c83e_failure() test_regression_invariant_P_B1_bc6627bc_failure() PASS	Resolved by PRs above. Some invariants are still failing because of other reasons.	Antonina Norair	Resolved	

		test_regression_invariant_P_B1_ba29ad51_failure()				
H04 PowerToken: User Balance is lost on each reset due to inflation roundingxz	High	On each reset via a PowerBootstrapToken, some coins are lost in the scaling down of the inflated supply. In theory, the rapid inflation of PowerToken should dilute this discrepancy out of having a meaningful effect on the system.		Michael De Luca	Won't fix, design	
M01 High Mint Ratio and High Collateral can cause Uint112 overflow in UpdateCollateral	Medium	Cap on Mint ratio	PR: https://github.com/MZero-Labs/protocol/pull/146 Commit in main: https://github.com/MZero-Labs/protocol/commit/f24d34a77d5c8082529df40f4c2f90587025f150	Michael De Luca	Resolved	Independent L01
M02 dynamic calculation of collateral expiry creates unintended consequences	Medium	UPDATE_COLLATERAL_INTERVAL can be updated to hurt minters or incentivize minters to avoid penalties by increasing interval		Antonina Norair	Won't fix, Design decision	
M03 resetToTokenHolders() functions will brick the new vote token if the bootstrap token's pastTotalSupply(epoch) returns 0	Medium	Check for bootstrap totalSupply != 0	PR: https://github.com/MZero-Labs/ttg/pull/236 Commit in main: https://github.com/MZero-Labs/ttg/commit/62290374c54d752a422ba559be52be64aea9c91a	Michael De Luca	Resolved	

M04 PowerToken: Account balances can exceed total supply.	Medium	It appears that `markParticipation` can cause user balances to exceed `totalSupply`. The following regression shows `Actor7` receives an extra `100` tokens (or 10%) after self delegating and getting a `markParticipation` call. It was not clear to us exactly where this happens or why and would strongly recommend further investigation. test_regression_invariant_P_B1_5cd1d968_failure()	test_regression_invariant_P_B1_5cd1d968_failure() Issue: C04 Account balances can exceed total supply Duplicate. Caused by the same bug as Prototech C01. Tests fails at line: https://gist.github.com/brianmcmichael/95a09be043d82d88a027b777dceb47e1#file-gistfile1-txt-L11 Call sequence lines: https://gist.github.com/brianmcmichael/95a09be043d82d88a027b777dceb47e1#file-gistfile1-txt-L716-L724 When markParticipation is called for Actor 7, the total supply is not increased as the voting power of Actor 7 is zero (because voting power has been given to address zero). But the balanceOf function, add unrealised inflation to the balance of Actor 7		Resolved	
M05 Invariant P_VD2 failure: Actor votes do not match delegated balance.	Medium	test_regression_invariant_P_VD2_dc8c60c1_failure In this regression, we arrive at a state where the actor has 1000 tokens delegated to them but 0 voting power in the Voting Epoch.	test_regression_invariant_P_VD2_dc8c60c1_failure() Issue: M05 Actor votes do not match delegated balance. Duplicate. This is also caused by the same bug as Prototech C01. Tests fails at the second last line: _powerTokenHandler.delegate(2, 1516140989270874076342658255876666786217000614717236199529083); Call sequence lines: https://gist.github.com/brianmcmichael/be5e9703d91fc94c99ca1774e3a84084#file-test_regression_inv		Resolved	

			ariant_p_vd2_dc8c60c1_failure-L1299-L1306			
L01 MToken updateIndex called multiple times in burn and mint	Low	The mint and burn functions on MToken are only callable by the MinterGateway. Inside , the functions that call mint and burn also call MinterGateway.updateIndex() which calls MToken.updateIndex() . Inside the MToken's mint and burn functions it checks if the recepient or account are earning and if they are it calls MToken.updateIndex().	Decoupling of indices can lead to described above situation. We acknowledge it, but don't see better ways to handle it without tight coupling of both indices	Michael De Luca	Acknowledged, design	
L02 cash token that doesn't return true on transfer	Low	Won't work for cash tokens being USDT and others that do not return true/false in transfers	Cash tokens will only be WETH and M itself.	Antonina Norair	Won't fix	
L03 updateCollateral potentially leaves the system in an undesirable state	Low	It was indicated that this should hold in your suggested invariants for minterGateway: collateralOf >= totalPendingRetrievals. However, by calling updateCollateral with a new, lower number and not passing any RetreivalIds, the minter would be able to create that exact situation. This violates a potential invariant: Sum of PendingRetreivals <= Sum of MinterState Collateral values.	The second proposed suggestion would not work in the case of a Minter with no owed M. *maxAllowedActiveOwedMOf* would return *0* since *totalPendingRetrievals_* will be greater than *collateral_* and *finalActiveOwedM_* will also return *0* since the Minter owed M balance is *0*. So the following check will be skipped and the retrieval created: *if (finalActiveOwedM_ > maxAllowedActiveOwedM_)* Explanation: https://github.com/MZero-Labs/protocol/pull/145#pullrequestreview-1908588872	Pierrick Turelier	Won't fix	

L04 proposeMint allows type(uint240).max, but mintM only allows type(uint112).max	Low	If all other requirements are met, proposeMint allows a user to propose minting up to type(uint240).max amount. However, such a proposal would overflow on type(uint112).max once called because of _getPresentAmountRoundedUp	We only enforce *uint240* in *proposeMint* and only *mintM* ensures that we won't overflow *uint112*.	Pierrick Turelier	Won't fix, design	
L05 TTG Setting Minter Rate too high will lead to updateIndex overflow	Low	Setting a rate too high for too long will result in multiplyIndices overflowing uint112 and lock the MinterGateway. As discussed you expect TTG to use rates between 0 and 40_000 for the minter rate, however, if governance deviates from this expectation or erroneously sets a very high rate, the system could be locked.	We are relying on governance setting sane values for mint ratio, rates, penalties etc.	Michael De Luca	Won't fix, design	
L06 TTG Set mintRatio() == 0 causes all positions to be undercollateralized	Low	Setting to 0 will cause all positions to be reported as under-collateralized causing all minters to be penalized on updateCollateral and reverts in proposeRetrieval, proposeMint, and mintM.		Antonina Norair	Won't fix, Design possibility Recommendation acknowledged	
L07 Invariant Violation/Accounting reported incorrectly		When a Minter is deactivated, their balances are wiped and they can't become active in the system anymore. However, the pendingCollateralRetrievalOf(minter, retreivalId) function will still return a value of past proposed retrievals. Recommendation: The easiest thing here would be to update pendingCollateralRetrievalOf so that it returns 0 for a deactivated minter	PR: https://github.com/MZero-Labs/protocol/pull/144 Commit in main: https://github.com/MZero-Labs/protocol/commit/0274c30c9f846d003012828ed034d0e4387b0992	Pierrick Turelier Antonina Norair	Resolved	

L08 Users can accidentally lock their funds	Low	It's a common problem that users lock their funds by sending them to the token contract directly or to address(0). For more context see this pull request in the Maker dss codebase on sending to the contract directly. While Maker chose not to prevent this behavior in DAI as it would have socialized the gas cost for this check to all users, they later regretted not putting the check in as it was a simple guard against extremely bad user experiences due to loss of funds.	PRs: https://github.com/MZero-Labs/common/pull/25 https://github.com/MZero-Labs/protocol/pull/147 https://github.com/MZero-Labs/ttg/pull/237 Commits in main: https://github.com/MZero-Labs/protocol/commit/ec4adf8a9a37da107c169f69aa45a57db49e7193 https://github.com/MZero-Labs/common/commit/0da7d2fecff44dc6d9d4c3d9b3cf6b8dd8926f9d https://github.com/MZero-Labs/ttg/commit/0c86903456266e3814d5086c63b680063d03bdb6	Michael De Luca Pierrick Turelier	Resolved	
L09 Inconsistent inflation due to rounding truncation	Low	Due to the small total supply and necessity to round down user balances across epochs, users with smaller balances will be unable to inflate past their initial balance, even when participating in an epoch.	Will be solved by experimenting and setting big enough initial supply.	Michael De Luca Antonina Norair	Won't fix	
I01 transferFrom with insufficient balance leads to Panic: over/underflow	Informational	If the `from` address does not have a sufficient balance to transfer to the `to` address, the operation will fail for `Panic: over/underflow`	PR: https://github.com/MZero-Labs/protocol/pull/138 Commit in main: https://github.com/MZero-Labs/protocol/commit/20b3b62f7d5dd97f5541f79cef51f8146a2dcbd5	Antonina Norair	Resolved	
I02 ERC20Extended: Insufficient allowance for transferFrom results in Panic underflow	Informational	Consider checking that the amount being transferred has an approval and provide a custom InsufficientAllowance() error.	PR: https://github.com/MZero-Labs/common/pull/16 Commit in main: https://github.com/MZero-Labs/common/commit/160d1058eab98ddb1e0406ae519c13f8b3d9674d	Antonina Norair	Resolved	

I03 can freeze deactivated minter	Informational	Unlike other functions that affect active minters, freezeMinter does not ensure that the minter has not already been deactivated. There does not seem to be an impact to this, other than the strange where the minter is both isDeactivated == true and has a value for their .	Situation is acknowledged and doesn't worth additional checks.	Pierrick Turelier Antonina Norair	Won't fix	
I04 StandardGovernor.t.sol does not test setKey	Informational	The StandardGovernor Unit test only checks that onlySelf test fails, it does not have a happy path test like the others. The setKey function is missing from the Mock as well.	In progress	Antonina Norair	Will be resolved next week, Low priority	
I05 Allow public reading of proposalFees in StandardGovernor	Informational	In order for the DistributionVault to get CashToken's to distribute, someone needs to call sendProposalFeeToVault , but there is no way on-chain to tell if there is a fee to send since _proposalFees is internal and does not have a public accessor. This could be a limiting UX for keepers to be able to distribute fees and maintain the system.	PR: https://github.com/MZero-Labs/ttg/pull/232 Commit in main: https://github.com/MZero-Labs/ttg/commit/aa5e183f903b905ceab8e9d79ec2e7aba8ce1184	Antonina Norair	Resolved	
I06 Reduce Duplicate Code to Prevent the Introduction of Bugs	Informational	The AuthorizationAlreadyUsed check on line 322 is the same as _revertIfAuthorizationAlreadyUsed	PR: https://github.com/MZero-Labs/common/pull/23 Commit in main: https://github.com/MZero-Labs/common/commit/2f8ea88fb653ee44beb29127d6112661317936c1	Pierrick Turelier	Resolved	
I07 SignatureChecker.sol vulnerable to signature malleability	Informational	The OZ libraries were found vulnerable to a signature malleability attack because they	We are aware of this issue and added a Natspec comment outlining the issue and recommending to not use the signature as a unique identifier.	Pierrick Turelier Michael De Luca	Acknowledged	

		allowed valid signatures for the same signed data. The MZero SignatureChecker.sol appears to implement the same validation pattern as the vulnerable OZ contracts. The vulnerability does not seem to affect the code at this time.	https://github.com/MZero-Labs/common/blob/4a37119f2da946c6d8ad7b9a70dfdd219225115b/src/libs/SignatureChecker.sol#L53			
I08 Investigate MinterGateway and MToken updateIndex	Informational		After creating a branch where _updateIndex was called at the beginning of the function and _updateRate was called at the end of the function, and having all conversion functions rely on _latestIndex rather than computing currentIndex(), the gas benefits were about 1% for MToken and 2-3% for MinterGateway, with the tradeoff being the code was harder to read and had “more lines”. We have decided not to make this change.	Michael De Luca	Acknowledged	
I09 MinterGateway does not validate that all signatures are in ascending order	Informational	If the threshold is met before checking all the signatures submitted, then subsequent signatures will not be checked that the validators were sorted correctly.	For the convenience of minter, we permit minters to send the minimum of valid signatures. Some of the signatures can come from not approved anymore validators. (ex validator was remove right before minter submitted tx)	Michael De Luca	Won’t fix, design	
I10 MinterGateway verifyValidatorSignatures could bail early	Informational	UpdateCollateral already verifies that validators_, timestamps_, signatures_ all have the same length. Verification could bail earlier by ensuring there are at least as many signatures as the required threshold.	Same as above	Michael De Luca	Won’t fix, design	
I11 OnBehalf -> OnBehalfOf	Informational	Names like allowEarningOnBehalf() don't match other nomenclature			Won’t fix, Method was deleted	

OpenZeppelin [22 total, report]

Issue	Severity [Declared, Suggested]	Description & Notes	Fix PR / commit	Owner	Status	Duplicate of
H01 H-01 Signature Replay Attack Possible in MinterGateway	High Low, not possible by design		PR: https://github.com/MZero-Labs/protocol/pull/114 Commit in Main: https://github.com/MZero-Labs/protocol/commit/b2c421c132cf6af6a18860ad17285b900be83163	Antonina Norair	Resolved	Three Sigma C01
M-01 Claiming of Distribution From DistributionVault Can Be Gamed	Medium Info			Antonina Norair	Won't fix	
M-02 M-Token and POWER Token Will Have Integration Issues on DEXs	Medium Info	The rebasing amounts of M-token or POWER token can be stolen from LP pools	We will create a wrapper contract to be able to use these tokens in LP pools.	Pierrick Turelier	Will implement	
M-03 DistributionVault Cannot Handle Rebasing Tokens	Medium Info	Rebasing amounts in the DistributionVault would be lost and not claimable by anyone	The DistributionVault will never be earning, so this isn't an issue.	Pierrick Turelier	Won't fix	
M-04 Rewards of an Epoch Are Claimable by Zero holders of a Future Epoch	Medium Info	Someone needs to call *distribute()* in order to account to token for distribution	As mentioned in the following comment, the incentive should be sufficient to incite someone to call *distribute()*	Pierrick Turelier	Won't fix	
M-05 Denial of Service Due to Transaction Running Out of Gas	Medium Info	Users that have not interacted with the protocol for a while may see their transaction revert because of out of gas errors	Epochs being 15 days long, this issue is more or less minimized. Also, users can reduce the interval between *startEpoch* and *endEpoch* if their transaction will	Pierrick Turelier	Won't fix	

			revert.			
M-06 Reward Tokens Can Get Stuck in the DistributionVault	Medium Info	Reward tokens sent to the DistributionVault will be unclaimable	We won't be sending these tokens to the DistributionVault, any users that do should be aware that their token will be locked	Pierrick Turelier	Won't fix	
L01 Input sanitization	Low	• The `transferFrom` function in the `ERC20Extended` library should check if `amount_ > spenderAllowance_` and revert with a meaningful error message instead of **reverting at the arithmetic underflow**. • The `_verifyValidatorSignatures` function in the `MinterGateway` contract should check if the **`threshold_`** is greater than 0. • The **`proposeMint`** function in `MinterGateway` contract does not validate that the `destination_` address is not `address(0)`. • The **`castVotes`**, **`castVotesBySig`**, and **`castVotesBySig`** functions in the `BatchGovernor` contract allow the user to cast votes on multiple proposals in single function call. These functions accept `proposalIds_` and `support_` arrays as user inputs but fail to check if the length of these arrays is the same. • According to the MZero Protocol Whitepaper, consider adding a check to the **constructor** of the Zero token which ascertains that the initial supply minted is equal to 1 billion.	1. Can add, works fine without 2. Feature, not bug. We want to allow threshold being 0 and do not require signatures 3. Added 4. Added 5. Invalid PRs: https://github.com/MZero-Labs/ttg/pull/222 https://github.com/MZero-Labs/common/pull/16 https://github.com/MZero-Labs/protocol/pull/131 Commits: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c https://github.com/MZero-Labs/common/commit/160d1058eab98ddb1e0406ae519c13f8b3d9674d https://github.com/MZero-Labs/protocol/commit/b40adad11e9a49d093138d0cb896f36fca6fe5ea	Antonina Norair	Resolved	

L02 L-02 Minter Disapproved by TTG Can Continue to Interact With the Protocol	Low Info		This is not an issue as it is intended behaviour. Respective minter actions will be practically infinitely more frequent than the frequency the minter's status can be toggled to inactive (iot can only happen once), so gas is saved by not having to query the Registrar for the minter's activation status every single time, and the minter's activation status is actually packed into a storage slot that will be accessed often during normal minter operations. A conscious decision made. Further, the account(s) that proposes and/or votes on and/or executes the proposal to deactivate a minter are just as incentivized to complete the last step by actually calling the deactivation function at the `MinterGateway`.	Antonina Norair Michael De Luca	Won't fix	
L03 Incorrect Comments	Low Info	• At **line 103** of the `IBatchGovernor` interface, the docstring above the **`castVotesBySig`** function states that a signer can cast votes via an arbitrary signature. However, this function allows anyone to cast a vote on behalf of the signer if they have a valid signature. • At **line 118** of the `StandardGovernor` contract, the comment wrongly asserts that standard proposals can be executed in epoch N+2, where N is the epoch in which the vote took place.	PR: https://github.com/MZero-Labs/ttg/pull/210 Commit in main: https://github.com/MZero-Labs/ttg/commit/a500f186cd06cd5971dc277631fb29c97ebc5431	Antonina Norair	1. Won't fix 2. Resolved	

L04 nextDeploy() Returns Incorrect Contract Address			I believe this issue is invalid since the suggestion appears wrong. First, applying the suggestion results in failing deployments and tests. Second, I believe the EVM does increment the nonce before using it to compute the address of a contract being deployed by a contract (i.e. the nonce 0 is never used).	Antonina Norair	Invalid issue, won't fix	
L05 EIP-1271 Signature Replay Attack Possible		Valid issue that would arise if EIP-1271 is not properly implemented by wallets. See: https://mirror.xyz/curiousapple.eth/pFqAdW2LiJ-6S4sg_u1z08k4vK6BCJ33LcyXpnNb8yU Bullet point 3: the signature can't be re-used since the nonce would be different.	PR: https://github.com/MZero-Labs/ttg/pull/248 Commit in main: https://github.com/MZero-Labs/ttg/commit/07e1ecaf6e324350453f4343b876ec45b3f857eb	Pierrick Turelier Michael De Luca	Resolved	
L06 L-06 Zero Amount of Power Tokens Can Be Minted	Low Info	Should be part of input sanitization. Duplicate. Add check to buy function for max and min amount == 0	PR: https://github.com/MZero-Labs/protocol/pull/131 https://github.com/MZero-Labs/common/pull/25 https://github.com/MZero-Labs/protocol/pull/147 https://github.com/MZero-Labs/ttg/pull/237 Commit in main: https://github.com/MZero-Labs/protocol/commit/b40adad11e9a49d093138d0cb896f36fca6fe5ea https://github.com/MZero-Labs/protocol/commit/a7925aa8cd5b4b36fd9f5c35cb947325d206f325 https://github.com/MZero-Labs/common/commit/0da7d2fecff44dc6d9d4c3d9b3cf6b8dd8926f9d https://github.com/MZero-Labs/ttg/commit/0c86903456266e3814d5086c63b680063d03bdb6	Antonina Norair Pierrick Turelier	Resolved	

L07 L-07 Zero Amount of M-tokens Can Be Minted or Burned	Low Info	Should be part of input sanitization. Duplicate.	PRs: https://github.com/MZero-Labs/protocol/pull/131 https://github.com/MZero-Labs/common/pull/25 https://github.com/MZero-Labs/protocol/pull/147 https://github.com/MZero-Labs/ttg/pull/237 Commits in main: https://github.com/MZero-Labs/protocol/commit/b40adad11e9a49d093138d0cb896f36fca6fe5ea https://github.com/MZero-Labs/protocol/commit/a7925aa8cd5b4b36fd9f5c35cb947325d206f325 https://github.com/MZero-Labs/common/commit/0da7d2fecff44dc6d9d4c3d9b3cf6b8dd8926f9d https://github.com/MZero-Labs/ttg/commit/0c86903456266e3814d5086c63b680063d03bdb6	Antonina Norair Pierrick Turelier	Resolved	
I01 Todo Comments in the Code	Info		PR: https://github.com/MZero-Labs/ttg/pull/222 Commit in main: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c	Antonina Norair	Resolved	
I02 Typographical Errors	Info	N-02 Typographical Errors	PR: https://github.com/MZero-Labs/ttg/pull/222 Commit in main: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c	Antonina Norair	Resolved	

I03 N-03 Lack of Security Contact	Info	Decentralized immutable protocol, no security contacts will be mentioned at launch.		Antonina Norair	Won't fix	
I04 N-04 TTG Is Not Fully Compatible With Community Tools	Info	Function signature stayed the same. It should be compatible with existent tools		Michael De Luca	Won't fix	
I05 N-05 Naming Suggestions	Info	Rename _getTotalSupply function in the EpochBasedVoteToken contract to _getTotalSupplyAtEpoch.	To be consistent with other names in the codebase, we will keep the original name.	Michael De Luca	Won't fix	
I06 N-06 Unused State Variables	Info	BatchGovernor can be used by community who can require quorumRatio		Antonina Norair	Won't fix	
I07 N-07 Unused Import	Info	Unused import in common	PR https://github.com/MZero-Labs/common/pull/16 Commit: https://github.com/MZero-Labs/common/commit/160d1058eab98ddb1e0406ae519c13f8b3d9674d	Antonina Norair	Resolved	
I08 N-08 Unused Function With internal Visibility	Info		PR: https://github.com/MZero-Labs/ttg/pull/222 Commit in main: https://github.com/MZero-Labs/ttg/commit/ead0c859f9f61d3347ad954288d0d15c268fa07c	Antonina Norair	Resolved	

Kirill Fedoseev [19 total, report]

Issue	Severity [Declared, Suggested]	Description & Notes	Fix PR / commit	Owner	Status	Duplicate of
H01 - Validator signatures can be double counted	High	Issue introduced while gas optimizing. Signature order check was bypassed if an unapproved validator sig was passed to the function.	PR: https://github.com/MZero-Labs/protocol/pull/136 Commit in main: https://github.com/MZero-Labs/protocol/commit/5e8a69d1a5b4af2c71c12da8b0b9a352f65a1ce4	Pierrick Turelier	Resolved	
M01 - Missing Approval event in permit implementation	Medium	*Approval* event is not emitted in *permit()*.	PR: https://github.com/MZero-Labs/common/pull/18 Commit in main: https://github.com/MZero-Labs/common/commit/81e204cb69ed8189b11102e5a661d63d55ef4858	Pierrick Turelier	Resolved	
[M-02] Validator signatures allow for retrieval amounts to be manipulated	Medium	Sync, adding one more parameter will lead to stack too deep error . Described situation is not valid in current reality Consider unique ids		Antonina Norair Michael De Luca Pierrick Turelier	Won't fix	
[L-01] Incorrect max cap for the mint ratio	Low		PR: https://github.com/MZero-Labs/protocol/pull/146 Commit in main: https://github.com/MZero-Labs/protocol/commit/f24d34a77d5c8082529df40f4c2f90587025f150	Michael De Luca	Resolved	

[L-02] Possible overflow in `convertToBasisPoints`	Low		PR: https://github.com/MZero-Labs/protocol/pull/143 Commit in main: https://github.com/MZero-Labs/protocol/commit/cf5463f665aeeaf4278cb783ca41791001621cff	Michael De Luca Antonina Norair	Resolved	Chainsecurity L11, L18
[L-03] Incorrect bound for return value of `_getUnrealizedInflation`	Low		PR: https://github.com/MZero-Labs/ttg/pull/233 Commit in main: https://github.com/MZero-Labs/ttg/commit/0c86903456266e3814d5086c63b680063d03bdb6	Michael De Luca Antonina Norair	Resolved	Three Sigma L05 ChainSecurity CS-MZEROCORE-014 Quantstamp H01
[L-04] Function `_castVotes` return value is inconsistent in `ThresholdGovernor`	Low	Functions from `IBatchGovernor` for casting multiple votes at once are supposed to return amount of cast votes. Natspec suggests that this amount is the same for all proposals. In `ThresholdGovernor`, however, this may not be the case if executed proposals were started during two different consecutive epochs. In such case, the return value is ambiguous	Cannot fix without either breaking the return standard or preventing users from batch voting on all active proposals (thus segregation by “started this epoch” and “started last epoch”). Further it would result in some code duplication and bulk. We won’t fix this as the issue is minor (only applies to contracts calling this function and using the return value, and not EOAs).	Michael De Luca		
[I-01] One of function `isValidECDSASignature` overloads is unused	Informational		PR: https://github.com/MZero-Labs/common/pull/20 Commit in main: https://github.com/MZero-Labs/common/commit/1212e8a32afda886d9497970c316f677cc0dc9cf	Pierrick Turelier	Resolved	Duplicate of Chainsecurity I01

[I-02] Inconsistent error handling in `_revertIfInvalidSignature`	Informational		PR: https://github.com/MZero-Labs/common/pull/20 Commit in main: https://github.com/MZero-Labs/common/commit/1212e8a32afda886d9497970c316f677cc0dc9cf	Pierrick Turelier	Resolved	
[I-03] ERC20 metadata storage for `name()` and `symbol()` can be made immutable	Informational		This approach would add code complexity and it is not really clear what the saving in term of gas is.	Pierrick Turelier	Won't fix, acknowledged	
[I-04] Empty implementation of `balanceOf` in `ERC20Extended`	Informational		PR: https://github.com/MZero-Labs/common/pull/24 Commit in main: https://github.com/MZero-Labs/common/commit/0da7d2fecff44dc6d9d4c3d9b3cf6b8dd8926f9d	Pierrick Turelier Michael De Luca	Resolved	
[I-05] Typos, errors and duplicates in NatSpec comments	Informational		PRs: https://github.com/MZero-Labs/protocol/pull/141 https://github.com/MZero-Labs/ttg/pull/227 Commits in main: https://github.com/MZero-Labs/protocol/commit/2b33a612e53bdddc2b1e5d089436d49f225327cf https://github.com/MZero-Labs/ttg/commit/6258d72fd4170531cdd238121c5045b74a8eaa65	Antonina Norair	Resolved	
[I-06] Redundant usage of `min40IgnoreZero`	Informational		PRs: https://github.com/MZero-Labs/protocol/pull/141 https://github.com/MZero-Labs/common/pull/19	Antonina Norair	Resolved	

			Commits in main: https://github.com/MZero-Labs/protocol/commit/2b33a612e53bdddc2b1e5d089436d49f225327cf https://github.com/MZero-Labs/common/commit/e323b1071cfc6082b0c4b42c31d1e4bce73eca89			
[I-07] Lack of key rotation mechanism for validators and minters	Informational	*Will be done in peripheral contracts with time* *removeFromAndAddToList can be used for this purpose too*		Antonina Norair	Won't fix, acknowledged	
[I-08] Redundant usage of `_getDefaultIfZero`	Informational	*Fixed*	PR: https://github.com/MZero-Labs/ttg/pull/250 Commit in main: https://github.com/MZero-Labs/ttg/commit/30857aed2309af558172e4f6d20506cf78d77fd2	Antonina Norair	Resolved	
[I-09] Proposal in `ThresholdGovernance` can be executed twice	Informational	*Michael De Luca is it possible? voteStart should be different* *sync*	It cannot happen, both conceptually and practically. I tried.	Michael De Luca	Acknowledged	
[I-10] Function `_castVote` for non-existent proposal reverts with different error	Informational		PR: https://github.com/MZero-Labs/ttg/pull/227 Commit in main: https://github.com/MZero-Labs/ttg/commit/6258d72fd4170531cdd238121c5045b74a8eaa65	Antonina Norair	Resolved	
[I-11] In *BALLOTS_WITH_REASON_TYPEHASH* the correct EIP-712 encoding for *string[]* should be the hash over concatenated list of string hashes, not over concatenation of string themselves.	Informational		PR: https://github.com/MZero-Labs/ttg/pull/245 Commit in main: https://github.com/MZero-Labs/ttg/commit/a0ff0a43d64121e7ec69d170b92a03d196bb88ec	Pierrick Turelier	Resolved	

[I-12] Various optimizations	Informational	*Redundant safe40 in https://github.com/MZero-Labs/protocol/blob/3e80bb1dd254ee43a2694b97a58e786c97d095e5/src/MinterGateway.sol#L1061 , as timestamp is already checked to be not greater than type(uint40).max* *Potential simplification for _getBalanceWithoutUnrealizedInflation*	PR: https://github.com/MZero-Labs/protocol/pull/154 Commit in main: https://github.com/MZero-Labs/protocol/commit/e28445a404dfa5d5549bf357b7d5b64f05cc2993 PR: https://github.com/MZero-Labs/ttg/pull/250 Commit in main: https://github.com/MZero-Labs/ttg/commit/30857aed2309af558172e4f6d20506cf78d77fd2	Antonina Norair	Resolved	

M^0 internal findings [6 total]

Issue	Description & Notes	Fix PR / commit	Owner	Status
currentIndex uint32(timestamp) overflow	If the elapsed timestamp computed in *currentIndex* overflows *uint32* the timestamp will overflow and wrap around. The index will then be computed improperly and will be lower than the expected index.	PR: https://github.com/MZero-Labs/protocol/pull/127	Pierrick Turelier	Canceled
Change max mint_ratio from 10_000% to 6_500% to avoid uint256 overflow		PR: https://github.com/MZero-Labs/protocol/pull/117 Commit in main: https://github.com/MZero-Labs/protocol/commit/06b6cb1593da5baffa60d50fb75a0767dd754a6b	Michael De Luca	Resolved
Delete unused getPastDelegates from ZERO token.		PR: https://github.com/MZero-Labs/ttg/pull/219 Commit in main: https://github.com/MZero-Labs/ttg/	Antonina Norair	Resolved

		commit/c543571550 81f91faa2cfbdaa60c51d34aa16442		
proposeRetrieval check that collateral != 0		PR: https://github.com/MZero-Labs/protocol/pull/131 Commit in main: https://github.com/MZero-Labs/protocol/commit/b40adad11e9a49d093138d0cb896f36fca6fe5ea	Antonina Norair	Resolved
Remove `startEarningOnBehalfOf` and `stopEarningOnBehalfOf`	Redundant use of optIn and optOut of earning	PR: https://github.com/MZero-Labs/protocol/pull/118 Commit in main: https://github.com/MZero-Labs/protocol/commit/E7c6daed554e7c188ade990d402d0aa8c02d4bf9	Michael De Luca	Resolved
Imporove index precision and overflows		PR: https://github.com/MZero-Labs/protocol/pull/148 https://github.com/MZero-Labs/protocol/pull/143	Michael De Luca	Resolved