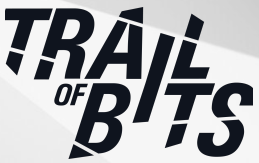

Legion Smart Contracts

Security Assessment (Summary Report)

August 13, 2024

Prepared for:
Legion

Prepared by: **Guillermo Larregay**

About Trail of Bits

Founded in 2012 and headquartered in New York, Trail of Bits provides technical security assessment and advisory services to some of the world's most targeted organizations. We combine high-end security research with a real-world attacker mentality to reduce risk and fortify code. With 100+ employees around the globe, we've helped secure critical software elements that support billions of end users, including Kubernetes and the Linux kernel.

We maintain an exhaustive list of publications at https://github.com/trailofbits/publications, with links to papers, presentations, public audit reports, and podcast appearances.

In recent years, Trail of Bits consultants have showcased cutting-edge research through presentations at CanSecWest, HCSS, Devcon, Empire Hacking, GrrCon, LangSec, NorthSec, the O'Reilly Security Conference, PyCon, REcon, Security BSides, and SummerCon.

We specialize in software testing and code review projects, supporting client organizations in the technology, defense, and finance industries, as well as government entities. Notable clients include HashiCorp, Google, Microsoft, Western Digital, and Zoom.

Trail of Bits also operates a center of excellence with regard to blockchain security. Notable projects include audits of Algorand, Bitcoin SV, Chainlink, Compound, Ethereum 2.0, MakerDAO, Matic, Uniswap, Web3, and Zcash.

To keep up to date with our latest news and announcements, please follow @trailofbits on Twitter and explore our public repositories at https://github.com/trailofbits. To engage us directly, visit our "Contact" page at https://www.trailofbits.com/contact, or email us at info@trailofbits.com.

Trail of Bits, Inc.
497 Carroll St., Space 71, Seventh Floor
Brooklyn, NY 11215
https://www.trailofbits.com
info@trailofbits.com

Notices and Remarks

Copyright and Distribution

Test Coverage Disclaimer

All activities undertaken by Trail of Bits in association with this project were performed in accordance with a statement of work and agreed upon project plan.

Security assessment projects are time-boxed and often reliant on information that may be provided by a client, its affiliates, or its partners. As a result, the findings documented in this report should not be considered a comprehensive list of security issues, flaws, or defects in the target system or codebase.

Trail of Bits uses automated testing techniques to rapidly test the controls and security properties of software. These techniques augment our manual security review work, but each has its limitations: for example, a tool may not generate a random edge case that violates a property or may not fully complete its analysis during the allotted time. Their use is also limited by the time and resource constraints of a project.

Table of Contents

Project Summary

Contact Information

The following project manager was associated with this project:

Mary O'Brien, Project Manager
mary.obrien@trailofbits.com

The following engineering director was associated with this project:

Josselin Feist, Engineering Director, Blockchain
josselin.feist@trailofbits.com

The following consultant was associated with this project:

Guillermo Larregay, Consultant
guillermo.larregay@trailofbits.com

Project Timeline

The significant events and milestones of the project are listed below.

Date	Event
June 6, 2024	Pre-project kickoff call
June 17, 2024	Delivery of report draft
June 17, 2024	Report readout meeting
June 25, 2024	Delivery of summary report
August 13, 2024	Delivery of summary report with fix review appendix

Project Targets

The engagement involved a review and testing of the following target.

Legion Smart Contracts

Repository	https://github.com/Legion-Team/legion/tree/master/contracts/www
Version	Commit `b0068bf`
Type	Solidity
Platform	EVM-compatible platforms

Executive Summary

Engagement Overview

Legion engaged Trail of Bits to review the security of the smart contracts that make up the Legion protocol. The protocol implements a way for independent projects to raise capital by selling tokens at a fixed price or by setting up sealed auctions in which the token price is unknown and set by the investors. Legion provides an on-chain centralized solution that complies with regulations, provides know-your-client (KYC) services, and handles the sale and auction of tokens.

One consultant conducted the review from June 10 to June 14, 2024, for a total of one engineer-week of effort. With full access to source code and documentation, we performed static and dynamic testing of the contracts in the codebase, using automated and manual processes.

Observations and Impact

The main point of concern for the audit was the flow of funds. Specifically, we sought to answer the following non-exhaustive list of questions:

- Can any party gain access to funds that do not belong to them?
- Can funds become stuck in a contract due to any party's either inaction or malicious action?
- Are time and access constraints enforced for all the auction stages?

The following are the contracts under audit, located in the `contracts/www/src/` directory:

- `LegionAddressRegistry.sol`
- `LegionKYCRegistry.sol`
- `LegionSealedBidAuction.sol`
- `LegionFixedPriceSale.sol`
- `LegionLinearVesting.sol`
- `LegionSealedBidAuctionFactory.sol`
- `LegionFixedPriceSaleFactory.sol`
- `LegionLinearVestingFactory.sol`

Because of the time-boxed nature of testing work, it is common to encounter coverage limitations. During this project, we were unable to perform comprehensive testing of the following system elements, which may warrant further review:

- **First- and third-party libraries and dependencies**
    - In particular, if the provided ECIES library has not been audited before, it should be reviewed through a cryptography-specific audit that also considers the particularities of blockchains.
- **Deployment scripts and private key management**
- **Networks other than the Ethereum main network**
    - Since the team plans to deploy contracts to other EVM-compatible networks, their differences with the Ethereum mainnet should be considered and analyzed before deployment.

In summary, the audit identified a total of 12 issues: two high-severity issues, three medium-severity issues, four informational-severity issues, and three issues of undetermined severity. Given that one high-severity issue and one medium-severity issue are related to the cryptographic parts of the sealed bids, it is recommended that Legion have an audit performed on the cryptography schemes and algorithms used.

Recommendations

Based on the codebase maturity evaluation and findings identified during the security review, Trail of Bits recommends that Legion take the following steps:

- **Remediate the findings resulting from this audit.** These findings should be addressed as part of a direct remediation or as part of any refactor that may occur when addressing other recommendations.
- **Refactor the auction and sale contracts.** In order to avoid code duplication and to minimize the chance of introducing issues, the sealed bid auction and fixed price sale contracts should be refactored. Refer to appendix F for more information.
- **Improve the test suite.** To ensure thorough test coverage, tests should be improved using the output of the mutation testing as guidance. Additionally, integration and fuzzing tests should be developed to increase the code robustness. Refer to appendix B for more information.

Codebase Maturity Evaluation

Trail of Bits uses a traffic-light protocol to provide each client with a clear understanding of the areas in which its codebase is mature, immature, or underdeveloped. Deficiencies identified here often stem from root causes within the software development life cycle that should be addressed through standardization measures (e.g., the use of common libraries, functions, or frameworks) or training and awareness programs.

Category	Summary	Result
Arithmetic	The codebase uses Solidity compiler version 0.8.25, and the arithmetic used is simple and easy to understand. No specific tests for arithmetic operations are present in the test suite provided.	**Satisfactory**
Auditing	All state-changing functions emit events, as they are used by the back end to function properly. However, the events are not used for security monitoring, and the team provided no incident response plan. Refer to appendix C for guidance on developing such a plan.	**Moderate**
Authentication / Access Controls	Three roles are defined within the system, and their privileges are documented. The team indicated that the `Legion` role is meant to be assigned to a multisignature wallet, but it is not possible to confirm this from the codebase; therefore, the risks cannot be evaluated. Refer to appendix D for recommendations related to multisignature wallets.	**Moderate**
Complexity Management	In general, functions have a clear and limited purpose, and parameter validation is performed for critical functions. However, there is significant code duplication between the auction and sale contracts, and the validation functions pass state variables as parameters, making the code harder to read and to understand.	**Moderate**
Decentralization	The system is centralized by design. The Legion team provides the encryption key pairs and Merkle tree roots for the protocol to work as intended, and the tokens collected as fees go to a protocol-controlled address.	**Weak**

	Some of the contracts are upgradeable. KYC is required for investors to receive the tokens they buy in auctions, which led to issue TOB-LGN-9.	
Documentation	The documentation available for the project consists of a README markdown file explaining how the protocol works, along with comments in the code files. It does not mention known risks or system limitations, and it does not include separate information for end users and developers that want to integrate with the protocol.	**Moderate**
Low-Level Manipulation	There are no low-level calls or assembly blocks in the code in scope.	**Strong**
Testing and Verification	The test coverage is not 100% for all files. Moreover, mutation testing revealed that not all contract functionality is tested in the current test suite. (See appendix B.) Only unit tests are available; no integration or fuzzing tests were provided by the team.	**Moderate**
Transaction Ordering	Some of the transactions in the mempool can be front-run and affect investors (TOB-LGN-7).	**Weak**

A. Code Maturity Categories

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

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

Rating Criteria	
Rating	**Description**
Strong	No issues were found, and the system exceeds industry standards.
Satisfactory	Minor issues were found, but the system is compliant with best practices.
Moderate	Some issues that may affect system safety were found.
Weak	Many issues that affect system safety were found.
Missing	A required component is missing, significantly affecting system safety.

Not Applicable	The category is not applicable to this review.
Not Considered	The category was not considered in this review.
Further Investigation Required	Further investigation is required to reach a meaningful conclusion.

B. Mutation Testing

This appendix outlines how we conducted mutation testing and highlights some of the most actionable results.

At a high level, mutation tests make several changes to each line of a target file and rerun the test suite against each change. Changes that result in test failures indicate adequate test coverage, while changes that do not result in test failures indicate gaps in test coverage. Mutation testing allows auditors to focus their review on areas of the codebase that are most likely to contain latent bugs, and it allows developers to identify and add missing tests.

We used an experimental new mutation tool, `slither-mutate`, to conduct our mutation testing campaign. This tool is currently part of our static analysis tool, Slither; it is custom-made for Solidity and features higher performance and fewer false positives than existing tools such as universalmutator.

The source-code mutation campaign was run against the in-scope smart contracts using the following command for the repository:

```
slither-mutate ./src  --test-cmd "forge test"
```

Figure B.1: A Bash command line that runs a mutation testing campaign against each Solidity file in the `src` directory

Additionally, the Necessist tool was used to mutate the tests provided by the team. This tool works similarly to `slither-mutate`, but instead of changing the code in the Solidity files, it mutates only test files. The command line used for this step is shown in figure B.2:

```
necessist ./test/*
```

Figure B.2: A Bash command line that runs a mutation testing campaign against each test file in the `test` directory

Consider the following notes about the above commands:

- The run time for the mutation campaigns depends on the number of files and tests that are part of the project and the time it takes to run the test suite. Moreover, some code statements can be eligible for several mutation rules, which increases total run time. In particular, when run on an M2 MacBook Pro, the `slither-mutate` campaign for the current codebase took around 7 hours to complete, and the Necessist campaign around 4 hours.

- In `slither-mutate`, the `--test-cmd` flag specifies the command to run to assess mutant validity. The additional `--fail-fast` or `--bail` flags will automatically be added to test commands to improve the run time.
- As with any automated tool, there can be false positives. Not every valid mutant indicates that a bug is present, but we strongly recommend manually reviewing all mutants, as they might indicate a lack of coverage or data validation in the tests. In particular, the following are some examples of false positives:
    - **Unsigned integer equivalences:** Some mutants in unsigned math yield the same output. For example, `var != 0` is equivalent to `var > 0`, and `var == 0` is equivalent to `var <= 0`.
    - **Bitwise operation equivalences:** For example, `var1 || 0` is equivalent to `var1 ^ 0`.
    - **Relaxing of function pure-ness:** Replacing the `pure` keyword with `view` usually causes a compiler warning but no error in tests.

Figures B.3 and B.4 show the output for the mutation testing campaigns on the repository.

```
INFO:Slither-Mutate:Starting mutation campaign in ./src
INFO:Slither-Mutate:Timing tests..
INFO:Slither-Mutate:Test suite passes in 12 seconds, commencing mutation campaign with a timeout of
24 seconds

INFO:Slither-Mutate:Mutating contract LegionFixedPriceSaleFactory
INFO:Slither-Mutate:[UOR] Line 39: '++totalInstances' ==> 'totalInstances++' --> UNCAUGHT
INFO:Slither-Mutate:Done mutating LegionFixedPriceSaleFactory.
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 4 (0.0%)
INFO:Slither-Mutate:Comment mutants: 0 uncaught of 2 (0.0%)
INFO:Slither-Mutate:Tweak mutants: 1 uncaught of 4 (25.0%)

INFO:Slither-Mutate:Mutating contract LegionFixedPriceSale
INFO:Slither-Mutate:[RR] Line 499: '_verifySaleResultsNotPublished(totalTokensAllocated)' ==>
'revert()' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 176: 'prefundStartTime = block.timestamp' ==> '//prefundStartTime =
block.timestamp' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 188: 'lockupEndTime = refundEndTime' ==> '//lockupEndTime =
refundEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 253: 'investorPositions[msg.sender].pledgedCapital = 0' ==>
'//investorPositions[msg.sender].pledgedCapital = 0' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 256: 'totalCapitalPledged -= amountToRefund' ==> '//totalCapitalPledged
-= amountToRefund' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 294: 'IERC20(bidToken).safeTransfer(msg.sender, (_totalCapitalRaised -
_legionFee))' ==> '//IERC20(bidToken).safeTransfer(msg.sender, (_totalCapitalRaised - _legionFee))'
--> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 305: 'askTokenAvailable' ==> '//askTokenAvailable' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 336: 'emit TokenAllocationClaimed(amount, msg.sender, vestingAddress)'
==> '//emit TokenAllocationClaimed(amount, msg.sender, vestingAddress)' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 364: 'investorPositions[msg.sender].pledgedCapital -= amount' ==>
'//investorPositions[msg.sender].pledgedCapital -= amount' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 367: 'totalCapitalPledged -= amount' ==> '//totalCapitalPledged -=
amount' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 370: 'emit ExcessCapitalClaimed(amount, msg.sender)' ==> '//emit
ExcessCapitalClaimed(amount, msg.sender)' --> UNCAUGHT
```

```
INFO:Slither-Mutate:[CR] Line 440: 'totalCapitalRaised = (tokensAllocated * tokenPrice) / (10 **
(ERC20(bidToken).decimals()))' ==> '//totalCapitalRaised = (tokensAllocated * tokenPrice) / (10 **
(ERC20(bidToken).decimals()))' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 523: 'investorPositions[msg.sender].pledgedCapital = 0' ==>
'//investorPositions[msg.sender].pledgedCapital = 0' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 526: 'totalCapitalPledged -= amountToClaim' ==> '//totalCapitalPledged
-= amountToClaim' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 177: 'prefundEndTime = prefundStartTime +
fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds' ==> 'prefundEndTime = prefundStartTime *
fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 178: 'startTime = prefundEndTime +
fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds' ==> 'startTime = prefundEndTime *
fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 179: 'endTime = startTime +
fixedPriceSalePeriodAndFeeConfig.salePeriodSeconds' ==> 'endTime = startTime *
fixedPriceSalePeriodAndFeeConfig.salePeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 191: 'lockupEndTime = endTime +
fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds' ==> 'lockupEndTime = endTime *
fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 288: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps +
_totalCapitalRaised) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 288: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps /
_totalCapitalRaised) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 288: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps %
_totalCapitalRaised) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 288: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) % 10000' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 218: 'totalCapitalPledged += amount' ==> 'totalCapitalPledged =
amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 218: 'totalCapitalPledged += amount' ==> 'totalCapitalPledged |=
amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 221: 'investorPositions[msg.sender].pledgedCapital += amount' ==>
'investorPositions[msg.sender].pledgedCapital = amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 221: 'investorPositions[msg.sender].pledgedCapital += amount' ==>
'investorPositions[msg.sender].pledgedCapital |= amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 221: 'investorPositions[msg.sender].pledgedCapital += amount' ==>
'investorPositions[msg.sender].pledgedCapital ^= amount' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 637: 'function _verifyValidConfig(
        FixedPriceSalePeriodAndFeeConfig calldata _fixedPriceSalePeriodAndFeeConfig,
        FixedPriceSaleAddressConfig calldata _fixedPriceSaleAddressConfig
    ) private pure ' ==> 'function _verifyValidConfig(
        FixedPriceSalePeriodAndFeeConfig calldata _fixedPriceSalePeriodAndFeeConfig,
        FixedPriceSaleAddressConfig calldata _fixedPriceSaleAddressConfig
    ) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 785: 'function _verifySaleResultsArePublished(uint256
_totalTokensAllocated) private pure ' ==> 'function _verifySaleResultsArePublished(uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 794: 'function _verifySaleResultsNotPublished(uint256
_totalTokensAllocated) private pure ' ==> 'function _verifySaleResultsNotPublished(uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 804: 'function _verifyCanSupplyTokens(uint256 _amount, uint256
_totalTokensAllocated) private pure ' ==> 'function _verifyCanSupplyTokens(uint256 _amount, uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 817: 'function _verifyCanPublishSaleResults(uint256
_totalTokensAllocated) private pure ' ==> 'function _verifyCanPublishSaleResults(uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 826: 'function _verifyCanPublishExcessCapitalResults(bytes32
_merkleRoot) private pure ' ==> 'function _verifyCanPublishExcessCapitalResults(bytes32 _merkleRoot)
private view ' --> UNCAUGHT
```

```
INFO:Slither-Mutate:[FHR] Line 835: 'function _verifySaleNotCanceled(bool _isCanceled) private pure '
==> 'function _verifySaleNotCanceled(bool _isCanceled) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 844: 'function _verifySaleIsCanceled(bool _isCanceled) private pure '
==> 'function _verifySaleIsCanceled(bool _isCanceled) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 853: 'function _verifyTokensNotSupplied(bool _tokensSupplied) private
pure ' ==> 'function _verifyTokensNotSupplied(bool _tokensSupplied) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 862: 'function _verifyTokensSupplied(bool _tokensSupplied) private
pure ' ==> 'function _verifyTokensSupplied(bool _tokensSupplied) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 643: '_fixedPriceSaleAddressConfig.bidToken == address(0)
                || _fixedPriceSaleAddressConfig.projectAdmin == address(0)
                || _fixedPriceSaleAddressConfig.legionAdmin == address(0)
                || _fixedPriceSaleAddressConfig.kycRegistry == address(0)
                || _fixedPriceSaleAddressConfig.vestingFactory == address(0)' ==>
'_fixedPriceSaleAddressConfig.bidToken == address(0)
                 &&  _fixedPriceSaleAddressConfig.projectAdmin == address(0)
                ' --> UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds == 0
                || _fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds == 0
                || _fixedPriceSalePeriodAndFeeConfig.salePeriodSeconds == 0
                || _fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds == 0
                || _fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds == 0
                || _fixedPriceSalePeriodAndFeeConfig.tokenPrice == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds == 0
                 &&  _fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds == 0
                ' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 136: 'askToken == address(0)' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==>
'!(fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds)' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 279: 'askToken != address(0)' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 297: '_legionFee != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 297: '_legionFee != 0' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 297: '_legionFee != 0' ==> '!(_legionFee != 0)' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 326: 'amount != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 362: 'amount != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 417: 'legionFee != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 417: 'legionFee != 0' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 417: 'legionFee != 0' ==> '!(legionFee != 0)' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 494: 'askToken != address(0)' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MVIE] Line 288: 'uint256 _legionFee = ' ==> 'uint256 _legionFee ' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==>
'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds  <
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==>
'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds  >
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==>
'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds  >=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==>
'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds  ==
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 185: 'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds <=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' ==>
```

```
'fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds  !=
fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 250: 'amountToRefund == 0' ==> 'amountToRefund  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 297: '_legionFee != 0' ==> '_legionFee  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 297: '_legionFee != 0' ==> '_legionFee  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 297: '_legionFee != 0' ==> '_legionFee  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 297: '_legionFee != 0' ==> '_legionFee  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 297: '_legionFee != 0' ==> '_legionFee  ==  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 326: 'amount != 0' ==> 'amount  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 326: 'amount != 0' ==> 'amount  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 362: 'amount != 0' ==> 'amount  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 362: 'amount != 0' ==> 'amount  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 408: 'legionFee != (legionFeeOnTokensSoldBps * amount) / 10000' ==>
'legionFee  <  (legionFeeOnTokensSoldBps * amount) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 417: 'legionFee != 0' ==> 'legionFee  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 417: 'legionFee != 0' ==> 'legionFee  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 417: 'legionFee != 0' ==> 'legionFee  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 417: 'legionFee != 0' ==> 'legionFee  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 417: 'legionFee != 0' ==> 'legionFee  ==  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 520: 'amountToClaim == 0' ==> 'amountToClaim  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 618: 'block.timestamp < _prefundEndTime' ==> 'block.timestamp  <=
_prefundEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 628: 'block.timestamp > _prefundEndTime' ==> 'block.timestamp  >=
_prefundEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 628: 'block.timestamp < _startTime' ==> 'block.timestamp  <=
_startTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.prefundPeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds
== 0' ==> '_fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds
== 0' ==> '_fixedPriceSalePeriodAndFeeConfig.prefundAllocationPeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.salePeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.salePeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.salePeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.salePeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.refundPeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.lockupPeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.tokenPrice == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.tokenPrice  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 652: '_fixedPriceSalePeriodAndFeeConfig.tokenPrice == 0' ==>
'_fixedPriceSalePeriodAndFeeConfig.tokenPrice  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 685: 'position.pledgedCapital == 0' ==> 'position.pledgedCapital  <=
0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 709: 'position.pledgedCapital == 0' ==> 'position.pledgedCapital  <=
0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 732: '_amount < 1 * 10 ** (ERC20(bidToken).decimals())' ==> '_amount
<=  1 * 10 ** (ERC20(bidToken).decimals())' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 741: 'block.timestamp < _endTime' ==> 'block.timestamp  <=  _endTime'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 750: 'block.timestamp > _endTime' ==> 'block.timestamp  >=  _endTime'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 759: 'block.timestamp < _refundEndTime' ==> 'block.timestamp  <=
_refundEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 768: 'block.timestamp > _refundEndTime' ==> 'block.timestamp  >=
_refundEndTime' --> UNCAUGHT
```

```
INFO:Slither-Mutate:[ROR] Line 777: 'block.timestamp < _lockupEndTime' ==> 'block.timestamp  <=
_lockupEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 786: '_totalTokensAllocated == 0' ==> '_totalTokensAllocated  <=  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 795: '_totalTokensAllocated != 0' ==> '_totalTokensAllocated  >  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 806: '_totalTokensAllocated == 0' ==> '_totalTokensAllocated  <=  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 809: '_amount != _totalTokensAllocated' ==> '_amount  >
_totalTokensAllocated' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 818: '_totalTokensAllocated != 0' ==> '_totalTokensAllocated  >  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 827: '_merkleRoot != bytes32(0)' ==> '_merkleRoot  >  bytes32(0)' -->
UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 328: 'address payable vestingAddress = _createVesting(
                msg.sender,
                uint64(vestingStartTime),
                uint64(vestingDurationSeconds),
                uint64(vestingCliffDurationSeconds)
            )' ==> 'address payable vestingAddress = _createVesting(
                msg.sender,
                uint32(vestingStartTime),
                uint32(vestingDurationSeconds),
                uint32(vestingCliffDurationSeconds)
            )' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 382: 'InvestorPosition memory position =
investorPositions[msg.sender]' ==> 'InvestorPosition storage position =
investorPositions[msg.sender]' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 676: 'InvestorPosition memory position = investorPositions[_investor]'
==> 'InvestorPosition storage position = investorPositions[_investor]' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 700: 'InvestorPosition memory position = investorPositions[_investor]'
==> 'InvestorPosition storage position = investorPositions[_investor]' --> UNCAUGHT
INFO:Slither-Mutate:Done mutating LegionFixedPriceSale.
INFO:Slither-Mutate:Revert mutants: 1 uncaught of 105 (0.9523809523809523%)
INFO:Slither-Mutate:Comment mutants: 13 uncaught of 113 (11.504424778761061%)
INFO:Slither-Mutate:Tweak mutants: 95 uncaught of 398 (23.86934673366834%)

INFO:Slither-Mutate:Mutating contract LegionLinearVestingFactory
INFO:Slither-Mutate:[CR] Line 57: 'emit NewLinearVestingCreated(beneficiary, startTimestamp,
durationSeconds, cliffDurationSeconds)' ==> '//emit NewLinearVestingCreated(beneficiary,
startTimestamp, durationSeconds, cliffDurationSeconds)' --> UNCAUGHT
INFO:Slither-Mutate:[UOR] Line 51: '++totalInstances' ==> 'totalInstances++' --> UNCAUGHT
INFO:Slither-Mutate:Done mutating LegionLinearVestingFactory.
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 6 (0.0%)
INFO:Slither-Mutate:Comment mutants: 1 uncaught of 8 (12.5%)
INFO:Slither-Mutate:Tweak mutants: 1 uncaught of 6 (16.666666666666668%)

INFO:Slither-Mutate:Mutating contract LegionAddressRegistry
INFO:Slither-Mutate:[MVIE] Line 29: 'address previousAddress = ' ==> 'address previousAddress ' -->
UNCAUGHT
INFO:Slither-Mutate:Done mutating LegionAddressRegistry.
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 3 (0.0%)
INFO:Slither-Mutate:Comment mutants: 0 uncaught of 3 (0.0%)
INFO:Slither-Mutate:Tweak mutants: 1 uncaught of 1 (100.0%)

INFO:Slither-Mutate:Mutating contract LegionSealedBidAuction
INFO:Slither-Mutate:[RR] Line 500: '_verifySaleResultsNotPublished(totalTokensAllocated)' ==>
'revert()' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 166: 'startTime = block.timestamp' ==> '//startTime = block.timestamp'
--> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 177: 'lockupEndTime = refundEndTime' ==> '//lockupEndTime =
refundEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 243: 'investorPositions[msg.sender].pledgedCapital = 0' ==>
'//investorPositions[msg.sender].pledgedCapital = 0' --> UNCAUGHT
```

```
INFO:Slither-Mutate:[CR] Line 246: 'totalCapitalPledged -= amountToRefund' ==> '//totalCapitalPledged
-= amountToRefund' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 284: 'IERC20(bidToken).safeTransfer(msg.sender, (_totalCapitalRaised -
_legionFee))' ==> '//IERC20(bidToken).safeTransfer(msg.sender, (_totalCapitalRaised - _legionFee))'
--> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 295: 'askTokenAvailable' ==> '//askTokenAvailable' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 326: 'emit TokenAllocationClaimed(amount, msg.sender, vestingAddress)'
==> '//emit TokenAllocationClaimed(amount, msg.sender, vestingAddress)' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 354: 'investorPositions[msg.sender].pledgedCapital -= amount' ==>
'//investorPositions[msg.sender].pledgedCapital -= amount' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 357: 'totalCapitalPledged -= amount' ==> '//totalCapitalPledged -=
amount' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 360: 'emit ExcessCapitalClaimed(amount, msg.sender)' ==> '//emit
ExcessCapitalClaimed(amount, msg.sender)' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 429: '_verifyCanPublishSaleResults(totalTokensAllocated)' ==>
'//_verifyCanPublishSaleResults(totalTokensAllocated)' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 441: 'totalCapitalRaised = capitalRaised' ==> '//totalCapitalRaised =
capitalRaised' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 524: 'investorPositions[msg.sender].pledgedCapital = 0' ==>
'//investorPositions[msg.sender].pledgedCapital = 0' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 527: 'totalCapitalPledged -= amountToClaim' ==> '//totalCapitalPledged
-= amountToClaim' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 167: 'endTime = startTime +
sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds' ==> 'endTime = startTime *
sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 180: 'lockupEndTime = endTime +
sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds' ==> 'lockupEndTime = endTime *
sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 278: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps +
_totalCapitalRaised) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 278: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps /
_totalCapitalRaised) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 278: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps %
_totalCapitalRaised) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[AOR] Line 278: 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) / 10000' ==> 'uint256 _legionFee = (legionFeeOnCapitalRaisedBps *
_totalCapitalRaised) % 10000' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 211: 'totalCapitalPledged += amount' ==> 'totalCapitalPledged =
amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 211: 'totalCapitalPledged += amount' ==> 'totalCapitalPledged |=
amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 214: 'investorPositions[msg.sender].pledgedCapital += amount' ==>
'investorPositions[msg.sender].pledgedCapital = amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 214: 'investorPositions[msg.sender].pledgedCapital += amount' ==>
'investorPositions[msg.sender].pledgedCapital |= amount' --> UNCAUGHT
INFO:Slither-Mutate:[ASOR] Line 214: 'investorPositions[msg.sender].pledgedCapital += amount' ==>
'investorPositions[msg.sender].pledgedCapital ^= amount' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 631: 'function _verifyValidConfig(
        SealedBidAuctionPeriodAndFeeConfig calldata _sealedBidAuctionPeriodAndFeeConfig,
        SealedBidAuctionAddressConfig calldata _sealedBidAuctionAddressConfig
    ) private pure ' ==> 'function _verifyValidConfig(
        SealedBidAuctionPeriodAndFeeConfig calldata _sealedBidAuctionPeriodAndFeeConfig,
        SealedBidAuctionAddressConfig calldata _sealedBidAuctionAddressConfig
    ) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 809: 'function _verifySaleResultsArePublished(uint256
_totalTokensAllocated) private pure ' ==> 'function _verifySaleResultsArePublished(uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 818: 'function _verifySaleResultsNotPublished(uint256
_totalTokensAllocated) private pure ' ==> 'function _verifySaleResultsNotPublished(uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
```

```
INFO:Slither-Mutate:[FHR] Line 828: 'function _verifyCanSupplyTokens(uint256 _amount, uint256
_totalTokensAllocated) private pure ' ==> 'function _verifyCanSupplyTokens(uint256 _amount, uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 841: 'function _verifyCanPublishSaleResults(uint256
_totalTokensAllocated) private pure ' ==> 'function _verifyCanPublishSaleResults(uint256
_totalTokensAllocated) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 850: 'function _verifyCanPublishExcessCapitalResults(bytes32
_merkleRoot) private pure ' ==> 'function _verifyCanPublishExcessCapitalResults(bytes32 _merkleRoot)
private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 859: 'function _verifySaleNotCanceled(bool _isCanceled) private pure '
==> 'function _verifySaleNotCanceled(bool _isCanceled) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 868: 'function _verifySaleIsCanceled(bool _isCanceled) private pure '
==> 'function _verifySaleIsCanceled(bool _isCanceled) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 877: 'function _verifyTokensNotSupplied(bool _tokensSupplied) private
pure ' ==> 'function _verifyTokensNotSupplied(bool _tokensSupplied) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 886: 'function _verifyTokensSupplied(bool _tokensSupplied) private
pure ' ==> 'function _verifyTokensSupplied(bool _tokensSupplied) private view ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 895: 'function _verifyPrivateKeyIsPublished(uint256 _privateKey)
private pure ' ==> 'function _verifyPrivateKeyIsPublished(uint256 _privateKey) private view ' -->
UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 637: '_sealedBidAuctionAddressConfig.bidToken == address(0)
                || _sealedBidAuctionAddressConfig.projectAdmin == address(0)
                || _sealedBidAuctionAddressConfig.legionAdmin == address(0)
                || _sealedBidAuctionAddressConfig.kycRegistry == address(0)
                || _sealedBidAuctionAddressConfig.vestingFactory == address(0)' ==>
'_sealedBidAuctionAddressConfig.bidToken == address(0)
                 &&  _sealedBidAuctionAddressConfig.projectAdmin == address(0)
                ' --> UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 646: '_sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds == 0
                || _sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds == 0
                || _sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds == 0' ==>
'_sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds == 0
                 &&  _sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds == 0
                ' --> UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 747: 'calcPubKey.x != publicKey.x || calcPubKey.y != publicKey.y' ==>
'calcPubKey.x != publicKey.x  &&  calcPubKey.y != publicKey.y' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 128: 'askToken == address(0)' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==>
'!(sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds)' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 269: 'askToken != address(0)' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 287: '_legionFee != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 287: '_legionFee != 0' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 287: '_legionFee != 0' ==> '!(_legionFee != 0)' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 316: 'amount != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 352: 'amount != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 407: 'legionFee != 0' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 407: 'legionFee != 0' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 407: 'legionFee != 0' ==> '!(legionFee != 0)' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 495: 'askToken != address(0)' ==> 'true' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 652: '!ECIES.isValid(_sealedBidAuctionPeriodAndFeeConfig.publicKey)'
==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 727: '!ECIES.isValid(_publicKey)' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 842: '_totalTokensAllocated != 0' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MVIE] Line 278: 'uint256 _legionFee = ' ==> 'uint256 _legionFee ' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==>
'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                 <  sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
```

```
INFO:Slither-Mutate:[ROR] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==>
'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                 >  sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==>
'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                 >=  sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==>
'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                 ==  sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 173: 'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                <= sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' ==>
'sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds
                 !=  sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 240: 'amountToRefund == 0' ==> 'amountToRefund  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 287: '_legionFee != 0' ==> '_legionFee  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 287: '_legionFee != 0' ==> '_legionFee  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 287: '_legionFee != 0' ==> '_legionFee  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 287: '_legionFee != 0' ==> '_legionFee  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 287: '_legionFee != 0' ==> '_legionFee  ==  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 316: 'amount != 0' ==> 'amount  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 316: 'amount != 0' ==> 'amount  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 352: 'amount != 0' ==> 'amount  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 352: 'amount != 0' ==> 'amount  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 398: 'legionFee != (legionFeeOnTokensSoldBps * amount) / 10000' ==>
'legionFee  <  (legionFeeOnTokensSoldBps * amount) / 10000' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 407: 'legionFee != 0' ==> 'legionFee  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 407: 'legionFee != 0' ==> 'legionFee  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 407: 'legionFee != 0' ==> 'legionFee  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 407: 'legionFee != 0' ==> 'legionFee  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 407: 'legionFee != 0' ==> 'legionFee  ==  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 521: 'amountToClaim == 0' ==> 'amountToClaim  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 646: '_sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds == 0' ==>
'_sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 646: '_sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds == 0' ==>
'_sealedBidAuctionPeriodAndFeeConfig.salePeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 646: '_sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds == 0'
==> '_sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 646: '_sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds == 0'
==> '_sealedBidAuctionPeriodAndFeeConfig.refundPeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 646: '_sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds == 0'
==> '_sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 646: '_sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds == 0'
==> '_sealedBidAuctionPeriodAndFeeConfig.lockupPeriodSeconds  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 679: 'position.pledgedCapital == 0' ==> 'position.pledgedCapital  <=
0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 703: 'position.pledgedCapital == 0' ==> 'position.pledgedCapital  <=
0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 731: 'keccak256(abi.encodePacked(_publicKey.x, _publicKey.y))
                != keccak256(abi.encodePacked(publicKey.x, publicKey.y))' ==>
'keccak256(abi.encodePacked(_publicKey.x, _publicKey.y))
                 <  keccak256(abi.encodePacked(publicKey.x, publicKey.y))' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 743: 'privateKey != 0' ==> 'privateKey  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 747: 'calcPubKey.x != publicKey.x' ==> 'calcPubKey.x  <  publicKey.x'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 747: 'calcPubKey.x != publicKey.x' ==> 'calcPubKey.x  >  publicKey.x'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 747: 'calcPubKey.y != publicKey.y' ==> 'calcPubKey.y  <  publicKey.y'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 747: 'calcPubKey.y != publicKey.y' ==> 'calcPubKey.y  >  publicKey.y'
--> UNCAUGHT
```

```
INFO:Slither-Mutate:[ROR] Line 756: '_amount < 1 * 10 ** (ERC20(bidToken).decimals())' ==> '_amount
<=  1 * 10 ** (ERC20(bidToken).decimals())' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 765: 'block.timestamp < _endTime' ==> 'block.timestamp  <=  _endTime'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 774: 'block.timestamp > _endTime' ==> 'block.timestamp  >=  _endTime'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 783: 'block.timestamp < _refundEndTime' ==> 'block.timestamp  <=
_refundEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 792: 'block.timestamp > _refundEndTime' ==> 'block.timestamp  >=
_refundEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 801: 'block.timestamp < _lockupEndTime' ==> 'block.timestamp  <=
_lockupEndTime' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 810: '_totalTokensAllocated == 0' ==> '_totalTokensAllocated  <=  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 819: '_totalTokensAllocated != 0' ==> '_totalTokensAllocated  >  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 830: '_totalTokensAllocated == 0' ==> '_totalTokensAllocated  <=  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 833: '_amount != _totalTokensAllocated' ==> '_amount  >
_totalTokensAllocated' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 842: '_totalTokensAllocated != 0' ==> '_totalTokensAllocated  <  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 842: '_totalTokensAllocated != 0' ==> '_totalTokensAllocated  >  0'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 851: '_merkleRoot != bytes32(0)' ==> '_merkleRoot  >  bytes32(0)' -->
UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 896: '_privateKey == 0' ==> '_privateKey  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 318: 'address payable vestingAddress = _createVesting(
                msg.sender,
                uint64(vestingStartTime),
                uint64(vestingDurationSeconds),
                uint64(vestingCliffDurationSeconds)
            )' ==> 'address payable vestingAddress = _createVesting(
                msg.sender,
                uint32(vestingStartTime),
                uint32(vestingDurationSeconds),
                uint32(vestingCliffDurationSeconds)
            )' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 372: 'InvestorPosition memory position =
investorPositions[msg.sender]' ==> 'InvestorPosition storage position =
investorPositions[msg.sender]' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 670: 'InvestorPosition memory position = investorPositions[_investor]'
==> 'InvestorPosition storage position = investorPositions[_investor]' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 694: 'InvestorPosition memory position = investorPositions[_investor]'
==> 'InvestorPosition storage position = investorPositions[_investor]' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 731: 'keccak256(abi.encodePacked(_publicKey.x, _publicKey.y))
                != keccak256(abi.encodePacked(publicKey.x, publicKey.y))' ==>
'keccak256(abi.encode(_publicKey.x, _publicKey.y))
                != keccak256(abi.encode(publicKey.x, publicKey.y))' --> UNCAUGHT
INFO:Slither-Mutate:Done mutating LegionSealedBidAuction.
INFO:Slither-Mutate:Revert mutants: 1 uncaught of 104 (0.9615384615384616%)
INFO:Slither-Mutate:Comment mutants: 14 uncaught of 114 (12.280701754385966%)
INFO:Slither-Mutate:Tweak mutants: 98 uncaught of 399 (24.56140350877193%)

INFO:Slither-Mutate:Mutating contract LegionLinearVesting
INFO:Slither-Mutate:[ROR] Line 27: 'block.timestamp < cliffEndTimestamp' ==> 'block.timestamp  <=
cliffEndTimestamp' --> UNCAUGHT
INFO:Slither-Mutate:Done mutating LegionLinearVesting.
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 6 (0.0%)
INFO:Slither-Mutate:Comment mutants: 0 uncaught of 8 (0.0%)
INFO:Slither-Mutate:Tweak mutants: 1 uncaught of 18 (5.555555555555555%)

INFO:Slither-Mutate:Mutating contract LegionKYCRegistry
INFO:Slither-Mutate:Done mutating LegionKYCRegistry.
```

```
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 3 (0.0%)
INFO:Slither-Mutate:Comment mutants: 0 uncaught of 3 (0.0%)
INFO:Slither-Mutate:Tweak mutants: 0 uncaught of 2 (0.0%)

INFO:Slither-Mutate:Mutating contract LegionSealedBidAuctionFactory
INFO:Slither-Mutate:[UOR] Line 39: '++totalInstances' ==> 'totalInstances++' --> UNCAUGHT
INFO:Slither-Mutate:Done mutating LegionSealedBidAuctionFactory.
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 4 (0.0%)
INFO:Slither-Mutate:Comment mutants: 0 uncaught of 2 (0.0%)
INFO:Slither-Mutate:Tweak mutants: 1 uncaught of 4 (25.0%)

INFO:Slither-Mutate:Mutating contract MockToken
INFO:Slither-Mutate:Done mutating MockToken.
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 1 (0.0%)
INFO:Slither-Mutate:Comment mutants: 0 uncaught of 1 (0.0%)
INFO:Slither-Mutate:Tweak mutants: 0 uncaught of 2 (0.0%)

INFO:Slither-Mutate:Mutating contract ECIES
INFO:Slither-Mutate:[CR] Line 33: 'return uint256(keccak256(abi.encodePacked(sharedSecret_, s1_)))'
==> '//return uint256(keccak256(abi.encodePacked(sharedSecret_, s1_)))' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 42: 'revert("Invalid public key.")' ==> '//revert("Invalid public
key.")' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 49: 'return p.x' ==> '//return p.x' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 91: 'messagePubKey_ = calcPubKey(Point(1, 2), privateKey_)' ==>
'//messagePubKey_ = calcPubKey(Point(1, 2), privateKey_)' --> UNCAUGHT
INFO:Slither-Mutate:[CR] Line 112: 'revert("Invalid generator point.")' ==> '//revert("Invalid
generator point.")' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 32: 'function deriveSymmetricKey(uint256 sharedSecret_, uint256 s1_)
public pure returns (uint256) ' ==> 'function deriveSymmetricKey(uint256 sharedSecret_, uint256 s1_)
public view returns (uint256) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 32: 'function deriveSymmetricKey(uint256 sharedSecret_, uint256 s1_)
public pure returns (uint256) ' ==> 'function deriveSymmetricKey(uint256 sharedSecret_, uint256 s1_)
private pure returns (uint256) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 32: 'function deriveSymmetricKey(uint256 sharedSecret_, uint256 s1_)
public pure returns (uint256) ' ==> 'function deriveSymmetricKey(uint256 sharedSecret_, uint256 s1_)
internal pure returns (uint256) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 37: 'function recoverSharedSecret(
        Point memory ciphertextPubKey_,
        uint256 privateKey_
    ) public view returns (uint256) ' ==> 'function recoverSharedSecret(
        Point memory ciphertextPubKey_,
        uint256 privateKey_
    ) private view returns (uint256) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 37: 'function recoverSharedSecret(
        Point memory ciphertextPubKey_,
        uint256 privateKey_
    ) public view returns (uint256) ' ==> 'function recoverSharedSecret(
        Point memory ciphertextPubKey_,
        uint256 privateKey_
    ) internal view returns (uint256) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 59: 'function decrypt(
        uint256 ciphertext_,
        Point memory ciphertextPubKey_,
        uint256 privateKey_,
        uint256 salt_
    ) public view returns (uint256 message_) ' ==> 'function decrypt(
        uint256 ciphertext_,
        Point memory ciphertextPubKey_,
        uint256 privateKey_,
        uint256 salt_
    ) internal view returns (uint256 message_) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 83: 'function encrypt(
        uint256 message_,
        Point memory recipientPubKey_,
```

```
        uint256 privateKey_,
        uint256 salt_
    ) public view returns (uint256 ciphertext_, Point memory messagePubKey_) ' ==> 'function encrypt(
        uint256 message_,
        Point memory recipientPubKey_,
        uint256 privateKey_,
        uint256 salt_
    ) internal view returns (uint256 ciphertext_, Point memory messagePubKey_) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 107: 'function calcPubKey(
        Point memory generator_,
        uint256 privateKey_
    ) public view returns (Point memory) ' ==> 'function calcPubKey(
        Point memory generator_,
        uint256 privateKey_
    ) internal view returns (Point memory) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 131: 'function isOnBn128(Point memory p) public pure returns (bool) '
==> 'function isOnBn128(Point memory p) private pure returns (bool) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 131: 'function isOnBn128(Point memory p) public pure returns (bool) '
==> 'function isOnBn128(Point memory p) internal pure returns (bool) ' --> UNCAUGHT
INFO:Slither-Mutate:[FHR] Line 137: 'function isValid(Point memory p) public pure returns (bool) '
==> 'function isValid(Point memory p) internal pure returns (bool) ' --> UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 45: 'privateKey_ >= GROUP_ORDER || privateKey_ == 0' ==> 'privateKey_
>= GROUP_ORDER  &&  privateKey_ == 0' --> UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 115: 'privateKey_ >= GROUP_ORDER || privateKey_ == 0' ==> 'privateKey_
>= GROUP_ORDER  &&  privateKey_ == 0' --> UNCAUGHT
INFO:Slither-Mutate:[LOR] Line 124: '!success || output.length == 0' ==> '!success  &&  output.length
== 0' --> UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 45: 'privateKey_ >= GROUP_ORDER || privateKey_ == 0' ==> 'false' -->
UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 115: 'privateKey_ >= GROUP_ORDER || privateKey_ == 0' ==> 'false' -->
UNCAUGHT
INFO:Slither-Mutate:[MIA] Line 124: '!success || output.length == 0' ==> 'false' --> UNCAUGHT
INFO:Slither-Mutate:[MVIE] Line 47: 'Point memory p = ' ==> 'Point memory p ' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 45: 'privateKey_ >= GROUP_ORDER' ==> 'privateKey_  >  GROUP_ORDER' -->
UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 45: 'privateKey_ >= GROUP_ORDER' ==> 'privateKey_  ==  GROUP_ORDER'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 45: 'privateKey_ == 0' ==> 'privateKey_  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 45: 'privateKey_ == 0' ==> 'privateKey_  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 115: 'privateKey_ >= GROUP_ORDER' ==> 'privateKey_  >  GROUP_ORDER'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 115: 'privateKey_ >= GROUP_ORDER' ==> 'privateKey_  ==  GROUP_ORDER'
--> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 115: 'privateKey_ == 0' ==> 'privateKey_  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 115: 'privateKey_ == 0' ==> 'privateKey_  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 124: 'output.length == 0' ==> 'output.length  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 124: 'output.length == 0' ==> 'output.length  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 133: '_fieldmul(p.y, p.y) == _fieldadd(_fieldmul(p.x, _fieldmul(p.x,
p.x)), 3)' ==> '_fieldmul(p.y, p.y)  <=  _fieldadd(_fieldmul(p.x, _fieldmul(p.x, p.x)), 3)' -->
UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 133: '_fieldmul(p.y, p.y) == _fieldadd(_fieldmul(p.x, _fieldmul(p.x,
p.x)), 3)' ==> '_fieldmul(p.y, p.y)  >=  _fieldadd(_fieldmul(p.x, _fieldmul(p.x, p.x)), 3)' -->
UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 1' ==> 'p.x  <  1' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 1' ==> 'p.x  >  1' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 1' ==> 'p.x  <=  1' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 1' ==> 'p.x  >=  1' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 1' ==> 'p.x  !=  1' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 2' ==> 'p.y  <  2' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 2' ==> 'p.y  >  2' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 2' ==> 'p.y  <=  2' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 2' ==> 'p.y  >=  2' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 2' ==> 'p.y  !=  2' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 0' ==> 'p.x  <  0' --> UNCAUGHT
```

```
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 0' ==> 'p.x  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 0' ==> 'p.x  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 0' ==> 'p.x  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.x == 0' ==> 'p.x  !=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 0' ==> 'p.y  <  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 0' ==> 'p.y  >  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 0' ==> 'p.y  <=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 0' ==> 'p.y  >=  0' --> UNCAUGHT
INFO:Slither-Mutate:[ROR] Line 138: 'p.y == 0' ==> 'p.y  !=  0' --> UNCAUGHT
INFO:Slither-Mutate:[SBR] Line 121: '(bool success, bytes memory output) =
           address(0x07).staticcall{gas: 6000}(abi.encode(p.x, p.y, scalar))' ==> '(bool success,
bytes memory output) =
           address(0x07).staticcall{gas: 6000}(abi.encodePacked(p.x, p.y, scalar))' --> UNCAUGHT
INFO:Slither-Mutate:Done mutating ECIES.
INFO:Slither-Mutate:Revert mutants: 0 uncaught of 9 (0.0%)
INFO:Slither-Mutate:Comment mutants: 5 uncaught of 15 (33.333333333333336%)
INFO:Slither-Mutate:Tweak mutants: 51 uncaught of 98 (52.04081632653061%)

INFO:Slither-Mutate:Done mutating ILegionLinearVestingFactory.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Done mutating ILegionLinearVesting.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Done mutating ILegionFixedPriceSale.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Done mutating ILegionKYCRegistry.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Done mutating ILegionSealedBidAuction.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Done mutating ILegionFixedPriceSaleFactory.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Done mutating ILegionSealedBidAuctionFactory.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Done mutating ILegionAddressRegistry.
INFO:Slither-Mutate:Zero Revert mutants analyzed
INFO:Slither-Mutate:Zero Comment mutants analyzed
INFO:Slither-Mutate:Zero Tweak mutants analyzed

INFO:Slither-Mutate:Finished mutation testing assessment of './src' in 7 hours
```

Figure B.3: Output from the source-code mutation testing campaign running `slither-mutate` in the repository

```
1726 candidates in 177 tests in 8 test files

test/LegionAddressRegistry.t.sol: dry running
test/LegionAddressRegistry.t.sol: mutilating
test/LegionAddressRegistry.t.sol:79:9-79:31: vm.prank(legionAdmin); passed

test/LegionFixedPriceSale.t.sol: dry running
test/LegionFixedPriceSale.t.sol: mutilating
test/LegionFixedPriceSale.t.sol:265:9-265:45: prepareCreateLegionFixedPriceSale(); passed
test/LegionFixedPriceSale.t.sol:279:9-300:11: setSaleConfig(
ILegionFixedPriceSale.FixedPriceSalePeriodAndFeeConfig({ prefundPeriodSeconds:
PREFUND_PERIOD_SECONDS, prefundAllocationPeriodSeconds: PREFUND_ALLOCATION_PERIOD_SECONDS,
salePeriodSeconds: SALE_PERIOD_SECONDS, refundPeriodSeconds: REFUND_PERIOD_SECONDS,
lockupPeriodSeconds: LOCKUP_PERIOD_SECONDS, vestingDurationSeconds: VESTING_DURATION_SECONDS,
vestingCliffDurationSeconds: VESTING_CLIFF_DURATION_SECONDS, legionFeeOnCapitalRaisedBps:
LEGION_FEE_CAPITAL_RAISED_BPS, legionFeeOnTokensSoldBps: LEGION_FEE_TOKENS_SOLD_BPS, tokenPrice:
TOKEN_PRICE }), ILegionFixedPriceSale.FixedPriceSaleAddressConfig({ bidToken: address(bidToken),
askToken: address(askToken), projectAdmin: projectAdmin, legionAdmin: legionAdmin, kycRegistry:
address(kycRegistry), vestingFactory: address(linearVestingFactory) }) ); passed
test/LegionFixedPriceSale.t.sol:318:9-339:11: setSaleConfig(
ILegionFixedPriceSale.FixedPriceSalePeriodAndFeeConfig({ prefundPeriodSeconds:
PREFUND_PERIOD_SECONDS, prefundAllocationPeriodSeconds: PREFUND_ALLOCATION_PERIOD_SECONDS,
salePeriodSeconds: SALE_PERIOD_SECONDS, refundPeriodSeconds: REFUND_PERIOD_SECONDS,
lockupPeriodSeconds: LOCKUP_PERIOD_SECONDS, vestingDurationSeconds: VESTING_DURATION_SECONDS,
vestingCliffDurationSeconds: VESTING_CLIFF_DURATION_SECONDS, legionFeeOnCapitalRaisedBps:
LEGION_FEE_CAPITAL_RAISED_BPS, legionFeeOnTokensSoldBps: LEGION_FEE_TOKENS_SOLD_BPS, tokenPrice:
TOKEN_PRICE }), ILegionFixedPriceSale.FixedPriceSaleAddressConfig({ bidToken: address(0), askToken:
address(0), projectAdmin: address(0), legionAdmin: address(0), kycRegistry: address(0),
vestingFactory: address(0) }) ); passed
test/LegionFixedPriceSale.t.sol:412:9-412:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionFixedPriceSale.t.sol:470:9-470:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionFixedPriceSale.t.sol:488:9-488:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionFixedPriceSale.t.sol:504:9-504:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionFixedPriceSale.t.sol:575:9-575:32: vm.warp(endTime() + 1); passed
test/LegionFixedPriceSale.t.sol:594:9-594:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionFixedPriceSale.t.sol:617:9-617:47: vm.warp(block.timestamp + 30 minutes); passed
test/LegionFixedPriceSale.t.sol:642:9-642:43: vm.warp(block.timestamp + 7 days); passed
test/LegionFixedPriceSale.t.sol:664:9-664:43: vm.warp(block.timestamp + 7 days); passed
test/LegionFixedPriceSale.t.sol:712:9-712:28: vm.warp(endTime()); passed
test/LegionFixedPriceSale.t.sol:735:9-735:43: vm.warp(block.timestamp + 7 days); passed
test/LegionFixedPriceSale.t.sol:763:9-763:43: vm.warp(block.timestamp + 7 days); passed
test/LegionFixedPriceSale.t.sol:779:9-779:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionFixedPriceSale.t.sol:781:9-781:43: vm.warp(block.timestamp + 7 days); passed
test/LegionFixedPriceSale.t.sol:829:9-829:38: vm.warp(refundEndTime() + 1); passed
test/LegionFixedPriceSale.t.sol:874:9-874:38: vm.warp(refundEndTime() - 1); passed
test/LegionFixedPriceSale.t.sol:895:9-895:38: vm.warp(endTime() + 1 weeks); passed
test/LegionFixedPriceSale.t.sol:943:9-943:32: vm.warp(endTime() + 1); passed
test/LegionFixedPriceSale.t.sol:992:9-992:32: vm.warp(endTime() - 1); passed
test/LegionFixedPriceSale.t.sol:1013:9-1013:38: vm.warp(endTime() + 1 weeks); passed
test/LegionFixedPriceSale.t.sol:1074:9-1074:37: vm.startPrank(projectAdmin); passed
test/LegionFixedPriceSale.t.sol:1075:9-1075:61: MockToken(askToken).mint(projectAdmin, 4090 * 1e18);
passed
test/LegionFixedPriceSale.t.sol:1076:9-1076:70: MockToken(askToken).approve(legionSaleInstance, 4090
* 1e18); passed
test/LegionFixedPriceSale.t.sol:1157:9-1157:40: vm.startPrank(nonProjectAdmin); passed
test/LegionFixedPriceSale.t.sol:1158:9-1158:65: MockToken(askToken).mint(nonProjectAdmin, 10250 *
1e18); passed
test/LegionFixedPriceSale.t.sol:1159:9-1159:71: MockToken(askToken).approve(legionSaleInstance, 10250
* 1e18); passed
test/LegionFixedPriceSale.t.sol:1188:9-1188:31: vm.prank(legionAdmin); passed
test/LegionFixedPriceSale.t.sol:1191:9-1191:37: vm.startPrank(projectAdmin); passed
test/LegionFixedPriceSale.t.sol:1192:9-1192:61: MockToken(askToken).mint(projectAdmin, 4100 * 1e18);
passed
```

```
test/LegionFixedPriceSale.t.sol:1193:9-1193:70: MockToken(askToken).approve(legionSaleInstance, 4100
* 1e18); passed
test/LegionFixedPriceSale.t.sol:1220:9-1220:37: vm.startPrank(projectAdmin); passed
test/LegionFixedPriceSale.t.sol:1221:9-1221:62: MockToken(askToken).mint(projectAdmin, 10240 * 1e18);
passed
test/LegionFixedPriceSale.t.sol:1222:9-1222:71: MockToken(askToken).approve(legionSaleInstance, 10240
* 1e18); passed
test/LegionFixedPriceSale.t.sol:1244:9-1244:38: vm.warp(refundEndTime() + 1); passed
test/LegionFixedPriceSale.t.sol:1246:9-1246:37: vm.startPrank(projectAdmin); passed
test/LegionFixedPriceSale.t.sol:1247:9-1247:62: MockToken(askToken).mint(projectAdmin, 10250 * 1e18);
passed
test/LegionFixedPriceSale.t.sol:1248:9-1248:71: MockToken(askToken).approve(legionSaleInstance, 10250
* 1e18); passed
test/LegionFixedPriceSale.t.sol:1314:9-1314:31: vm.prank(legionAdmin); passed
test/LegionFixedPriceSale.t.sol:1368:9-1368:31: vm.prank(legionAdmin); passed
test/LegionFixedPriceSale.t.sol:1395:9-1395:38: vm.warp(lockupEndTime() - 1); passed
test/LegionFixedPriceSale.t.sol:1507:9-1507:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1540:9-1540:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1563:9-1563:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1583:9-1583:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1585:9-1585:38: vm.warp(refundEndTime() - 1); passed
test/LegionFixedPriceSale.t.sol:1603:9-1603:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1626:9-1626:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1630:9-1630:35: vm.prank(nonProjectAdmin); passed
test/LegionFixedPriceSale.t.sol:1683:9-1683:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1687:9-1687:116: excessClaimProofInvestor2[0] =
bytes32(0xe6ec166fcb24e8b45dbf44e2137a36706ae07288095a733f7439bb2f81a94051); passed
test/LegionFixedPriceSale.t.sol:1688:9-1688:116: excessClaimProofInvestor2[1] =
bytes32(0x61c19f281f94212e62b60d017ca806d139d4f0da454abbc73e9533e0d99f398b); passed
test/LegionFixedPriceSale.t.sol:1690:9-1690:32: vm.warp(endTime() - 1); passed
test/LegionFixedPriceSale.t.sol:1708:9-1708:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1712:9-1712:116: excessClaimProofInvestor2[0] =
bytes32(0xe6ec166fcb24e8b45dbf44e2137a36706ae07288095a733f7439bb2f81a94051); passed
test/LegionFixedPriceSale.t.sol:1713:9-1713:116: excessClaimProofInvestor2[1] =
bytes32(0x61c19f281f94212e62b60d017ca806d139d4f0da454abbc73e9533e0d99f398b); passed
test/LegionFixedPriceSale.t.sol:1739:9-1739:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1743:9-1743:116: excessClaimProofInvestor2[0] =
bytes32(0xe6ec166fcb24e8b45dbf44e2137a36706ae07288095a733f7439bb2f81a94052); passed
test/LegionFixedPriceSale.t.sol:1744:9-1744:116: excessClaimProofInvestor2[1] =
bytes32(0x61c19f281f94212e62b60d017ca806d139d4f0da454abbc73e9533e0d99f398c); passed
test/LegionFixedPriceSale.t.sol:1797:9-1797:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:1849:9-1849:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:1879:9-1879:30: prepareKYCRegistry(); passed
test/LegionFixedPriceSale.t.sol:1880:9-1880:26: prepareVesting(); passed
test/LegionFixedPriceSale.t.sol:1885:9-1885:43: kycProofInvestor4[0] = bytes32(0); passed
test/LegionFixedPriceSale.t.sol:1886:9-1886:43: kycProofInvestor4[1] = bytes32(0); passed
test/LegionFixedPriceSale.t.sol:1888:9-1888:110: claimProofInvestor4[0] =
bytes32(0x80b8f5ec23ca02728bd0a7df5dc15da7274522d13d5a8a1f918d1c7bad3306a8); passed
test/LegionFixedPriceSale.t.sol:1889:9-1889:110: claimProofInvestor4[1] =
bytes32(0xa2144e298b31c1e3aa896eab357fd937fb7a574cc7237959b432e96a9423492c); passed
test/LegionFixedPriceSale.t.sol:1900:9-1900:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:1903:9-1903:38: vm.warp(lockupEndTime() + 1); passed
test/LegionFixedPriceSale.t.sol:1925:9-1925:26: prepareVesting(); passed
test/LegionFixedPriceSale.t.sol:1945:9-1945:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:1948:9-1948:38: vm.warp(lockupEndTime() - 1); passed
test/LegionFixedPriceSale.t.sol:1970:9-1970:26: prepareVesting(); passed
test/LegionFixedPriceSale.t.sol:1978:9-1978:110: claimProofInvestor2[0] =
bytes32(0x2054afa66e2c4ccd7ade9889c78d8cf4a46f716980dafb935d11ce1e564fa39c); passed
test/LegionFixedPriceSale.t.sol:1979:9-1979:110: claimProofInvestor2[1] =
bytes32(0xa2144e298b31c1e3aa896eab357fd937fb7a574cc7237959b432e96a9423492c); passed
```

```
test/LegionFixedPriceSale.t.sol:1990:9-1990:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:1993:9-1993:38: vm.warp(lockupEndTime() + 1); passed
test/LegionFixedPriceSale.t.sol:2015:9-2015:26: prepareVesting(); passed
test/LegionFixedPriceSale.t.sol:2035:9-2035:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:2038:9-2038:38: vm.warp(lockupEndTime() + 1); passed
test/LegionFixedPriceSale.t.sol:2080:9-2080:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:2110:9-2110:26: prepareVesting(); passed
test/LegionFixedPriceSale.t.sol:2128:9-2128:35: vm.prank(nonProjectAdmin); passed
test/LegionFixedPriceSale.t.sol:2149:9-2149:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionFixedPriceSale.t.sol:2150:9-2150:30: prepareKYCRegistry(); passed
test/LegionFixedPriceSale.t.sol:2151:9-2151:26: prepareVesting(); passed
test/LegionFixedPriceSale.t.sol:2156:9-2156:108: kycProofInvestor2[0] =
bytes32(0x607bca8f7c1c56da874da29cd62c3769b9880d38a258a91fc6dd1cfb6b4d1a8e); passed
test/LegionFixedPriceSale.t.sol:2157:9-2157:108: kycProofInvestor2[1] =
bytes32(0x64db0af4e3097c2974bf8abb17133c058f2076a7074b7aecfa02f9a04f6ccfa0); passed
test/LegionFixedPriceSale.t.sol:2159:9-2159:110: claimProofInvestor2[0] =
bytes32(0x2054afa66e2c4ccd7ade9889c78d8cf4a46f716980dafb935d11ce1e564fa39c); passed
test/LegionFixedPriceSale.t.sol:2160:9-2160:110: claimProofInvestor2[1] =
bytes32(0xa2144e298b31c1e3aa896eab357fd937fb7a574cc7237959b432e96a9423492c); passed
test/LegionFixedPriceSale.t.sol:2162:9-2162:38: vm.warp(lockupEndTime() + 1); passed
test/LegionFixedPriceSale.t.sol:2206:9-2206:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:2235:9-2235:30: prepareKYCRegistry(); passed
test/LegionFixedPriceSale.t.sol:2236:9-2236:26: prepareVesting(); passed
test/LegionFixedPriceSale.t.sol:2241:9-2241:108: kycProofInvestor2[0] =
bytes32(0x607bca8f7c1c56da874da29cd62c3769b9880d38a258a91fc6dd1cfb6b4d1a8e); passed
test/LegionFixedPriceSale.t.sol:2242:9-2242:108: kycProofInvestor2[1] =
bytes32(0x64db0af4e3097c2974bf8abb17133c058f2076a7074b7aecfa02f9a04f6ccfa0); passed
test/LegionFixedPriceSale.t.sol:2244:9-2244:110: claimProofInvestor2[0] =
bytes32(0xa840fc41b720079e7bce186d116e4412058ec6b29d3a5722a1f377be1e0d0992); passed
test/LegionFixedPriceSale.t.sol:2245:9-2245:110: claimProofInvestor2[1] =
bytes32(0x78158beefdfe129d958b15ef9bd1674f0aa97f179110ab76222f24f31db73155); passed
test/LegionFixedPriceSale.t.sol:2256:9-2256:69:
ILegionFixedPriceSale(legionSaleInstance).withdrawCapital(); passed
test/LegionFixedPriceSale.t.sol:2259:9-2259:71: vm.warp(lockupEndTime() +
VESTING_CLIFF_DURATION_SECONDS + 1); passed

test/LegionFixedPriceSaleFactory.t.sol: dry running
test/LegionFixedPriceSaleFactory.t.sol: mutilating
test/LegionFixedPriceSaleFactory.t.sol:149:9-170:11: setSaleConfig(
ILegionFixedPriceSale.FixedPriceSalePeriodAndFeeConfig({ prefundPeriodSeconds:
PREFUND_PERIOD_SECONDS, prefundAllocationPeriodSeconds: PREFUND_ALLOCATION_PERIOD_SECONDS,
salePeriodSeconds: SALE_PERIOD_SECONDS, refundPeriodSeconds: REFUND_PERIOD_SECONDS,
lockupPeriodSeconds: LOCKUP_PERIOD_SECONDS, vestingDurationSeconds: VESTING_DURATION_SECONDS,
vestingCliffDurationSeconds: VESTING_CLIFF_DURATION_SECONDS, legionFeeOnCapitalRaisedBps:
LEGION_FEE_CAPITAL_RAISED_BPS, legionFeeOnTokensSoldBps: LEGION_FEE_TOKENS_SOLD_BPS, tokenPrice:
TOKEN_PRICE }), ILegionFixedPriceSale.FixedPriceSaleAddressConfig({ bidToken: address(0), askToken:
address(0), projectAdmin: address(0), legionAdmin: address(0), kycRegistry: address(0),
vestingFactory: address(0) }) ); passed

test/LegionKYCRegistry.t.sol: dry running
test/LegionKYCRegistry.t.sol: mutilating
test/LegionKYCRegistry.t.sol:88:9-88:42: kycProofInvestor[0] = bytes32(0); passed
test/LegionKYCRegistry.t.sol:89:9-89:42: kycProofInvestor[1] = bytes32(0); passed

test/LegionLinearVesting.t.sol: dry running
test/LegionLinearVesting.t.sol: mutilating
test/LegionLinearVesting.t.sol:71:9-71:44: prepareCreateLegionLinearVesting(); passed

test/LegionLinearVestingFactory.t.sol: dry running
test/LegionLinearVestingFactory.t.sol: mutilating
```

```
test/LegionSealedBidAuction.t.sol: dry running
test/LegionSealedBidAuction.t.sol: mutilating
test/LegionSealedBidAuction.t.sol:303:9-303:47: prepareCreateLegionSealedBidAuction(); passed
test/LegionSealedBidAuction.t.sol:319:9-338:11: setSaleConfig(
ILegionSealedBidAuction.SealedBidAuctionPeriodAndFeeConfig({ salePeriodSeconds: SALE_PERIOD_SECONDS,
refundPeriodSeconds: REFUND_PERIOD_SECONDS, lockupPeriodSeconds: LOCKUP_PERIOD_SECONDS,
vestingDurationSeconds: VESTING_DURATION_SECONDS, vestingCliffDurationSeconds:
VESTING_CLIFF_DURATION_SECONDS, legionFeeOnCapitalRaisedBps: LEGION_FEE_CAPITAL_RAISED_BPS,
legionFeeOnTokensSoldBps: LEGION_FEE_TOKENS_SOLD_BPS, publicKey: PUBLIC_KEY }),
ILegionSealedBidAuction.SealedBidAuctionAddressConfig({ bidToken: address(bidToken), askToken:
address(askToken), projectAdmin: projectAdmin, legionAdmin: legionAdmin, kycRegistry:
address(kycRegistry), vestingFactory: address(linearVestingFactory) }) ); passed
test/LegionSealedBidAuction.t.sol:356:9-375:11: setSaleConfig(
ILegionSealedBidAuction.SealedBidAuctionPeriodAndFeeConfig({ salePeriodSeconds: SALE_PERIOD_SECONDS,
refundPeriodSeconds: REFUND_PERIOD_SECONDS, lockupPeriodSeconds: LOCKUP_PERIOD_SECONDS,
vestingDurationSeconds: VESTING_DURATION_SECONDS, vestingCliffDurationSeconds:
VESTING_CLIFF_DURATION_SECONDS, legionFeeOnCapitalRaisedBps: LEGION_FEE_CAPITAL_RAISED_BPS,
legionFeeOnTokensSoldBps: LEGION_FEE_TOKENS_SOLD_BPS, publicKey: PUBLIC_KEY }),
ILegionSealedBidAuction.SealedBidAuctionAddressConfig({ bidToken: address(0), askToken: address(0),
projectAdmin: address(0), legionAdmin: address(0), kycRegistry: address(0), vestingFactory:
address(0) }) ); passed
test/LegionSealedBidAuction.t.sol:450:9-450:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionSealedBidAuction.t.sol:469:9-469:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionSealedBidAuction.t.sol:471:9-471:32: vm.warp(endTime() + 1); passed
test/LegionSealedBidAuction.t.sol:488:9-488:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionSealedBidAuction.t.sol:505:9-505:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionSealedBidAuction.t.sol:579:9-579:32: vm.warp(endTime() + 1); passed
test/LegionSealedBidAuction.t.sol:597:9-597:32: prepareSealedBidData(); passed
test/LegionSealedBidAuction.t.sol:599:9-599:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionSealedBidAuction.t.sol:623:9-623:47: vm.warp(block.timestamp + 30 minutes); passed
test/LegionSealedBidAuction.t.sol:649:9-649:43: vm.warp(block.timestamp + 7 days); passed
test/LegionSealedBidAuction.t.sol:672:9-672:43: vm.warp(block.timestamp + 7 days); passed
test/LegionSealedBidAuction.t.sol:724:9-724:28: vm.warp(endTime()); passed
test/LegionSealedBidAuction.t.sol:748:9-748:43: vm.warp(block.timestamp + 7 days); passed
test/LegionSealedBidAuction.t.sol:777:9-777:43: vm.warp(block.timestamp + 7 days); passed
test/LegionSealedBidAuction.t.sol:792:9-792:32: prepareSealedBidData(); passed
test/LegionSealedBidAuction.t.sol:794:9-794:47: prepareMintAndApproveInvestorTokens(); passed
test/LegionSealedBidAuction.t.sol:796:9-796:43: vm.warp(block.timestamp + 7 days); passed
test/LegionSealedBidAuction.t.sol:848:9-848:38: vm.warp(refundEndTime() + 1); passed
test/LegionSealedBidAuction.t.sol:901:9-901:38: vm.warp(refundEndTime() + 1); passed
test/LegionSealedBidAuction.t.sol:925:9-925:38: vm.warp(refundEndTime() - 1); passed
test/LegionSealedBidAuction.t.sol:949:9-949:38: vm.warp(endTime() + 1 weeks); passed
test/LegionSealedBidAuction.t.sol:1001:9-1001:32: vm.warp(endTime() + 1); passed
test/LegionSealedBidAuction.t.sol:1052:9-1052:32: vm.warp(endTime() - 1); passed
test/LegionSealedBidAuction.t.sol:1074:9-1074:38: vm.warp(endTime() + 1 weeks); passed
test/LegionSealedBidAuction.t.sol:1141:9-1141:37: vm.startPrank(projectAdmin); passed
test/LegionSealedBidAuction.t.sol:1142:9-1142:61: MockToken(askToken).mint(projectAdmin, 4090 *
1e18); passed
test/LegionSealedBidAuction.t.sol:1143:9-1143:82:
MockToken(askToken).approve(legionSealedBidAuctionInstance, 4090 * 1e18); passed
test/LegionSealedBidAuction.t.sol:1229:9-1229:40: vm.startPrank(nonProjectAdmin); passed
test/LegionSealedBidAuction.t.sol:1230:9-1230:65: MockToken(askToken).mint(nonProjectAdmin, 10250 *
1e18); passed
test/LegionSealedBidAuction.t.sol:1231:9-1231:83:
MockToken(askToken).approve(legionSealedBidAuctionInstance, 10250 * 1e18); passed
test/LegionSealedBidAuction.t.sol:1263:9-1263:31: vm.prank(legionAdmin); passed
test/LegionSealedBidAuction.t.sol:1266:9-1266:37: vm.startPrank(projectAdmin); passed
test/LegionSealedBidAuction.t.sol:1267:9-1267:61: MockToken(askToken).mint(projectAdmin, 4100 *
1e18); passed
test/LegionSealedBidAuction.t.sol:1268:9-1268:82:
MockToken(askToken).approve(legionSealedBidAuctionInstance, 4100 * 1e18); passed
test/LegionSealedBidAuction.t.sol:1298:9-1298:37: vm.startPrank(projectAdmin); passed
```

```
test/LegionSealedBidAuction.t.sol:1299:9-1299:62: MockToken(askToken).mint(projectAdmin, 10240 *
1e18); passed
test/LegionSealedBidAuction.t.sol:1300:9-1300:83:
MockToken(askToken).approve(legionSealedBidAuctionInstance, 10240 * 1e18); passed
test/LegionSealedBidAuction.t.sol:1325:9-1325:38: vm.warp(refundEndTime() + 1); passed
test/LegionSealedBidAuction.t.sol:1327:9-1327:37: vm.startPrank(projectAdmin); passed
test/LegionSealedBidAuction.t.sol:1328:9-1328:62: MockToken(askToken).mint(projectAdmin, 10250 *
1e18); passed
test/LegionSealedBidAuction.t.sol:1329:9-1329:83:
MockToken(askToken).approve(legionSealedBidAuctionInstance, 10250 * 1e18); passed
test/LegionSealedBidAuction.t.sol:1401:9-1401:31: vm.prank(legionAdmin); passed
test/LegionSealedBidAuction.t.sol:1461:9-1461:31: vm.prank(legionAdmin); passed
test/LegionSealedBidAuction.t.sol:1491:9-1491:38: vm.warp(lockupEndTime() - 1); passed
test/LegionSealedBidAuction.t.sol:1609:9-1609:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1645:9-1645:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1671:9-1671:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1692:9-1692:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1694:9-1694:38: vm.warp(refundEndTime() - 1); passed
test/LegionSealedBidAuction.t.sol:1713:9-1713:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1737:9-1737:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1741:9-1741:35: vm.prank(nonProjectAdmin); passed
test/LegionSealedBidAuction.t.sol:1799:9-1799:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1803:9-1803:116: excessClaimProofInvestor2[0] =
bytes32(0xe6ec166fcb24e8b45dbf44e2137a36706ae07288095a733f7439bb2f81a94051); passed
test/LegionSealedBidAuction.t.sol:1804:9-1804:116: excessClaimProofInvestor2[1] =
bytes32(0x61c19f281f94212e62b60d017ca806d139d4f0da454abbc73e9533e0d99f398b); passed
test/LegionSealedBidAuction.t.sol:1806:9-1806:32: vm.warp(endTime() - 1); passed
test/LegionSealedBidAuction.t.sol:1827:9-1827:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1831:9-1831:116: excessClaimProofInvestor2[0] =
bytes32(0xe6ec166fcb24e8b45dbf44e2137a36706ae07288095a733f7439bb2f81a94051); passed
test/LegionSealedBidAuction.t.sol:1832:9-1832:116: excessClaimProofInvestor2[1] =
bytes32(0x61c19f281f94212e62b60d017ca806d139d4f0da454abbc73e9533e0d99f398b); passed
test/LegionSealedBidAuction.t.sol:1861:9-1861:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1865:9-1865:116: excessClaimProofInvestor2[0] =
bytes32(0xe6ec166fcb24e8b45dbf44e2137a36706ae07288095a733f7439bb2f81a94052); passed
test/LegionSealedBidAuction.t.sol:1866:9-1866:116: excessClaimProofInvestor2[1] =
bytes32(0x61c19f281f94212e62b60d017ca806d139d4f0da454abbc73e9533e0d99f398c); passed
test/LegionSealedBidAuction.t.sol:1927:9-1927:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:1986:9-1986:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2017:9-2017:30: prepareKYCRegistry(); passed
test/LegionSealedBidAuction.t.sol:2018:9-2018:26: prepareVesting(); passed
test/LegionSealedBidAuction.t.sol:2023:9-2023:43: kycProofInvestor4[0] = bytes32(0); passed
test/LegionSealedBidAuction.t.sol:2024:9-2024:43: kycProofInvestor4[1] = bytes32(0); passed
test/LegionSealedBidAuction.t.sol:2026:9-2026:110: claimProofInvestor4[0] =
bytes32(0x80b8f5ec23ca02728bd0a7df5dc15da7274522d13d5a8a1f918d1c7bad3306a8); passed
test/LegionSealedBidAuction.t.sol:2027:9-2027:110: claimProofInvestor4[1] =
bytes32(0xa2144e298b31c1e3aa896eab357fd937fb7a574cc7237959b432e96a9423492c); passed
test/LegionSealedBidAuction.t.sol:2040:9-2040:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2043:9-2043:38: vm.warp(lockupEndTime() + 1); passed
test/LegionSealedBidAuction.t.sol:2066:9-2066:26: prepareVesting(); passed
test/LegionSealedBidAuction.t.sol:2088:9-2088:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2091:9-2091:38: vm.warp(lockupEndTime() - 1); passed
test/LegionSealedBidAuction.t.sol:2114:9-2114:26: prepareVesting(); passed
test/LegionSealedBidAuction.t.sol:2122:9-2122:110: claimProofInvestor2[0] =
bytes32(0x2054afa66e2c4ccd7ade9889c78d8cf4a46f716980dafb935d11ce1e564fa39c); passed
test/LegionSealedBidAuction.t.sol:2123:9-2123:110: claimProofInvestor2[1] =
bytes32(0xa2144e298b31c1e3aa896eab357fd937fb7a574cc7237959b432e96a9423492c); passed
test/LegionSealedBidAuction.t.sol:2136:9-2136:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2139:9-2139:38: vm.warp(lockupEndTime() + 1); passed
test/LegionSealedBidAuction.t.sol:2162:9-2162:26: prepareVesting(); passed
```

```
test/LegionSealedBidAuction.t.sol:2184:9-2184:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2187:9-2187:38: vm.warp(lockupEndTime() + 1); passed
test/LegionSealedBidAuction.t.sol:2232:9-2232:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2263:9-2263:26: prepareVesting(); passed
test/LegionSealedBidAuction.t.sol:2283:9-2283:35: vm.prank(nonProjectAdmin); passed
test/LegionSealedBidAuction.t.sol:2305:9-2305:49: preparePledgedCapitalFromAllInvestors(); passed
test/LegionSealedBidAuction.t.sol:2306:9-2306:30: prepareKYCRegistry(); passed
test/LegionSealedBidAuction.t.sol:2307:9-2307:26: prepareVesting(); passed
test/LegionSealedBidAuction.t.sol:2312:9-2312:108: kycProofInvestor2[0] =
bytes32(0x607bca8f7c1c56da874da29cd62c3769b9880d38a258a91fc6dd1cfb6b4d1a8e); passed
test/LegionSealedBidAuction.t.sol:2313:9-2313:108: kycProofInvestor2[1] =
bytes32(0x64db0af4e3097c2974bf8abb17133c058f2076a7074b7aecfa02f9a04f6ccfa0); passed
test/LegionSealedBidAuction.t.sol:2315:9-2315:110: claimProofInvestor2[0] =
bytes32(0x2054afa66e2c4ccd7ade9889c78d8cf4a46f716980dafb935d11ce1e564fa39c); passed
test/LegionSealedBidAuction.t.sol:2316:9-2316:110: claimProofInvestor2[1] =
bytes32(0xa2144e298b31c1e3aa896eab357fd937fb7a574cc7237959b432e96a9423492c); passed
test/LegionSealedBidAuction.t.sol:2318:9-2318:38: vm.warp(lockupEndTime() + 1); passed
test/LegionSealedBidAuction.t.sol:2365:9-2365:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2395:9-2395:30: prepareKYCRegistry(); passed
test/LegionSealedBidAuction.t.sol:2396:9-2396:26: prepareVesting(); passed
test/LegionSealedBidAuction.t.sol:2401:9-2401:108: kycProofInvestor2[0] =
bytes32(0x607bca8f7c1c56da874da29cd62c3769b9880d38a258a91fc6dd1cfb6b4d1a8e); passed
test/LegionSealedBidAuction.t.sol:2402:9-2402:108: kycProofInvestor2[1] =
bytes32(0x64db0af4e3097c2974bf8abb17133c058f2076a7074b7aecfa02f9a04f6ccfa0); passed
test/LegionSealedBidAuction.t.sol:2404:9-2404:110: claimProofInvestor2[0] =
bytes32(0xa840fc41b720079e7bce186d116e4412058ec6b29d3a5722a1f377be1e0d0992); passed
test/LegionSealedBidAuction.t.sol:2405:9-2405:110: claimProofInvestor2[1] =
bytes32(0x78158beefdfe129d958b15ef9bd1674f0aa97f179110ab76222f24f31db73155); passed
test/LegionSealedBidAuction.t.sol:2418:9-2418:83:
ILegionSealedBidAuction(legionSealedBidAuctionInstance).withdrawCapital(); passed
test/LegionSealedBidAuction.t.sol:2421:9-2421:71: vm.warp(lockupEndTime() +
VESTING_CLIFF_DURATION_SECONDS + 1); passed
test/LegionSealedBidAuction.t.sol:2490:9-2490:38: vm.warp(refundEndTime() - 1); passed

test/LegionSealedBidAuctionFactory.t.sol: dry running
test/LegionSealedBidAuctionFactory.t.sol: mutilating
test/LegionSealedBidAuctionFactory.t.sol:148:9-167:11: setSaleConfig(
ILegionSealedBidAuction.SealedBidAuctionPeriodAndFeeConfig({ salePeriodSeconds: SALE_PERIOD_SECONDS,
refundPeriodSeconds: REFUND_PERIOD_SECONDS, lockupPeriodSeconds: LOCKUP_PERIOD_SECONDS,
vestingDurationSeconds: VESTING_DURATION_SECONDS, vestingCliffDurationSeconds:
VESTING_CLIFF_DURATION_SECONDS, legionFeeOnCapitalRaisedBps: LEGION_FEE_CAPITAL_RAISED_BPS,
legionFeeOnTokensSoldBps: LEGION_FEE_TOKENS_SOLD_BPS, publicKey: PUBLIC_KEY }),
ILegionSealedBidAuction.SealedBidAuctionAddressConfig({ bidToken: address(0), askToken: address(0),
projectAdmin: address(0), legionAdmin: address(0), kycRegistry: address(0), vestingFactory:
address(0) }) ); passed
```

Figure B.4: Output from the test mutation testing campaign running `necessist` in the repository

The line in figure B.5 shows an example of lack of coverage in the test suite, as there is no specific test that checks for this particular situation. Commenting out the variable assignment makes the test suite pass because there is no test case that checks that the pledged amount is effectively set to zero after the function is run.

```
INFO:Slither-Mutate:Mutating contract LegionSealedBidAuction
[...]
```

```
INFO:Slither-Mutate:[CR] Line 243: 'investorPositions[msg.sender].pledgedCapital =
0' ==> '//investorPositions[msg.sender].pledgedCapital = 0' --> UNCAUGHT
```

Figure B.5: An example of a mutant showing lack of test coverage

On the other hand, the line in figure B.6 is an example of a mutant that does not directly indicate issues in the test suite because it highlights equivalent code or harmless code. Since the value of `totalInstances` is not used again in the same line, both variants are equivalent.

```
INFO:Slither-Mutate:Mutating contract LegionLinearVestingFactory
[...]
INFO:Slither-Mutate:[UOR] Line 51: '++totalInstances' ==> 'totalInstances++' -->
UNCAUGHT
```

Figure B.6: An example of a mutant showing a false positive

We recommend improving the test suite given the valid mutants found by `slither-mutate` and `necessist` and rerunning a mutation test campaign using the command lines shown in figures B.1 and B.2. This will improve test coverage and prevent potential issues caused by specific situations that are currently not tested.

C. Incident Response Recommendations

This section provides recommendations on formulating an incident response plan.

- **Identify the parties (either specific people or roles) responsible for implementing the mitigations when an issue occurs (e.g., deploying smart contracts, pausing contracts, upgrading the front end, etc.).**
- **Document internal processes for addressing situations in which a deployed remedy does not work or introduces a new bug.**
    - Consider documenting a plan of action for handling failed remediations.
- **Clearly describe the intended contract deployment process.**
- **Outline the circumstances under which Legion will compensate users affected by an issue (if any).**
    - Issues that warrant compensation could include an individual or aggregate loss or a loss resulting from user error, a contract flaw, or a third-party contract flaw.
- **Document how the team plans to stay up to date on new issues that could affect the system; awareness of such issues will inform future development work and help the team secure the deployment toolchain and the external on-chain and off-chain services that the system relies on.**
    - Identify sources of vulnerability news for each language and component used in the system, and subscribe to updates from each source. Consider creating a private Discord channel in which a bot will post the latest vulnerability news; this will provide the team with a way to track all updates in one place. Lastly, consider assigning certain team members to track news about vulnerabilities in specific system components.
- **Determine when the team will seek assistance from external parties (e.g., auditors, affected users, other protocol developers) and how it will onboard them.**
    - Effective remediation of certain issues may require collaboration with external parties.
- **Define contract behavior that would be considered abnormal by off-chain monitoring solutions.**

It is best practice to perform periodic dry runs of scenarios outlined in the incident response plan to find omissions and opportunities for improvement and to develop “muscle memory.” Additionally, document the frequency with which the team should perform dry runs of various scenarios, and perform dry runs of more likely scenarios more regularly. Create a template to be filled out with descriptions of any necessary improvements after each dry run.

D. Security Best Practices for Using Multisignature Wallets

Consensus requirements for sensitive actions, such as spending funds from a wallet, are meant to mitigate the risks of the following:

- Any one person overruling the judgment of others
- Failures caused by any one person's mistake
- Failures caused by the compromise of any one person's credentials

For example, in a 2-of-3 multisignature wallet, the authority to execute a "spend" transaction would require a consensus of two individuals in possession of two of the wallet's three private keys. For this model to be useful, the following conditions are required:

1. The private keys must be stored or held separately, and access to each one must be limited to a unique individual.
2. If the keys are physically held by third-party custodians (e.g., a bank), multiple keys should not be stored with the same custodian. (Doing so would violate requirement #1.)
3. The person asked to provide the second and final signature on a transaction (i.e., the cosigner) should refer to a preestablished policy specifying the conditions for approving the transaction by signing it with his or her key.
4. The cosigner should also verify that the half-signed transaction was generated willingly by the intended holder of the first signature's key.

Requirement #3 prevents the cosigner from becoming merely a "deputy" acting on behalf of the first signer (forfeiting the decision-making responsibility to the first signer and defeating the security model). If the cosigner can refuse to approve the transaction for any reason, the due-diligence conditions for approval may be unclear. That is why a policy for validating transactions is needed. A verification policy could include the following:

- A protocol for handling a request to cosign a transaction (e.g., a half-signed transaction will be accepted only via an approved channel)
- An allowlist of specific addresses allowed to be the payee of a transaction
- A limit on the amount of funds spent in a single transaction or in a single day

Requirement #4 mitigates the risks associated with a single stolen key. For example, say that an attacker somehow acquired the unlocked Ledger Nano S of one of the signatories. A voice call from the cosigner to the initiating signatory to confirm the transaction would reveal that the key had been stolen and that the transaction should not be cosigned. If the signatory were under an active threat of violence, he or she could use a duress code (a code word, a phrase, or another signal agreed upon in advance) to covertly alert the others that the transaction had not been initiated willingly, without alerting the attacker.

E. Token Integration Checklist

The following checklist provides recommendations for interactions with arbitrary tokens. Every unchecked item should be justified and its associated risks understood. For an up-to-date version of the checklist, see `crytic/building-secure-contracts`.

For convenience, all Slither utilities can be run directly on a token address, such as the following:

```
slither-check-erc 0xdac17f958d2ee523a2206206994597c13d831ec7 TetherToken --erc erc20
slither-check-erc 0x06012c8cf97BEaD5deAe237070F9587f8E7A266d KittyCore --erc erc721
```

To follow this checklist, use the following output from Slither for the token:

```
slither-check-erc [target] [contractName] [optional: --erc ERC_NUMBER]
slither [target] --print human-summary
slither [target] --print contract-summary
slither-prop . --contract ContractName # requires configuration, and use of Echidna
and Manticore
```

General Considerations

- ❏ **The contract has a security review.** Avoid interacting with contracts that lack a security review. Check the length of the assessment (i.e., the level of effort), the reputation of the security firm, and the number and severity of the findings.
- ❏ **You have contacted the developers.** You may need to alert their team to an incident. Look for appropriate contacts on `blockchain-security-contacts`.
- ❏ **They have a security mailing list for critical announcements.** Their team should advise users when critical issues are found or when upgrades occur.

Contract Composition

- ❏ **The contract avoids unnecessary complexity.** The token should be a simple contract; a token with complex code requires a higher standard of review. Use Slither's `human-summary` printer to identify complex code.
- ❏ **The contract uses `SafeMath` or Solidity 0.8.0+.** Contracts that do not use `SafeMath` require a higher standard of review. Inspect the contract by hand for `SafeMath`/Solidity 0.8.0+ usage.
- ❏ **The contract has only a few non-token-related functions.** Non-token-related functions increase the likelihood of an issue in the contract. Use Slither's `contract-summary` printer to broadly review the code used in the contract.

- ❏ **The token has only one address.** Tokens with multiple entry points for balance updates can break internal bookkeeping based on the address (e.g., `balances[token_address][msg.sender]` may not reflect the actual balance).

Owner Privileges

- ❏ **The token is not upgradeable.** Upgradeable contracts may change their rules over time. Use Slither's `human-summary` printer to determine whether the contract is upgradeable.
- ❏ **The owner has limited minting capabilities.** Malicious or compromised owners can misuse minting capabilities. Use Slither's `human-summary` printer to review minting capabilities, and consider manually reviewing the code.
- ❏ **The token is not pausable.** Malicious or compromised owners can trap contracts relying on pausable tokens. Identify pausable code by hand.
- ❏ **The owner cannot denylist the contract.** Malicious or compromised owners can trap contracts relying on tokens with a denylist. Identify denylisting features by hand.
- ❏ **The team behind the token is known and can be held responsible for misuse.** Contracts with anonymous development teams or teams that reside in legal shelters require a higher standard of review.

ERC-20 Tokens

ERC-20 Conformity Checks

Slither includes a utility, `slither-check-erc`, that reviews the conformance of a token to many related ERC standards. Use `slither-check-erc` to review the following:

- ❏ **`Transfer` and `transferFrom` return a Boolean.** Several tokens do not return a Boolean on these functions. As a result, their calls in the contract might fail.
- ❏ **The `name`, `decimals`, and `symbol` functions are present if used.** These functions are optional in the ERC-20 standard and may not be present.
- ❏ **`Decimals` returns a `uint8`.** Several tokens incorrectly return a `uint256`. In such cases, ensure that the value returned is less than 255.
- ❏ **The token mitigates the known ERC-20 race condition.** The ERC-20 standard has a known ERC-20 race condition that must be mitigated to prevent attackers from stealing tokens.

Slither includes a utility, `slither-prop`, that generates unit tests and security properties that can discover many common ERC flaws. Use `slither-prop` to review the following:

- ❏ **The contract passes all unit tests and security properties from `slither-prop`.** Run the generated unit tests and then check the properties with Echidna and Manticore.

Risks of ERC-20 Extensions

The behavior of certain contracts may differ from the original ERC specification. Conduct a manual review of the following conditions:

- ❏ **The token is not an ERC-777 token and has no external function call in `transfer` or `transferFrom`.** External calls in the transfer functions can lead to reentrancies.
- ❏ **`Transfer` and `transferFrom` should not take a fee.** Deflationary tokens can lead to unexpected behavior.
- ❏ **Potential interest earned from the token is accounted for.** Some tokens distribute interest to token holders. This interest may be trapped in the contract if not accounted for.

Token Scarcity

Reviews of token scarcity issues must be executed manually. Check for the following conditions:

- ❏ **The supply is owned by more than a few users.** If a few users own most of the tokens, they can influence operations based on the tokens' repartition.
- ❏ **The total supply is sufficient.** Tokens with a low total supply can be easily manipulated.
- ❏ **The tokens are in more than a few exchanges.** If all the tokens are in one exchange, a compromise of the exchange could compromise the contract relying on the token.
- ❏ **Users understand the risks associated with a large amount of funds or flash loans.** Contracts relying on the token balance must account for attackers with a large amount of funds or attacks executed through flash loans.
- ❏ **The token does not allow flash minting.** Flash minting can lead to substantial swings in the balance and the total supply, which necessitate strict and comprehensive overflow checks in the operation of the token.

F. Code Quality Findings

This appendix contains findings that do not have immediate or obvious security implications. However, they may facilitate exploit chains targeting other vulnerabilities, become easily exploitable in future releases, or decrease code readability. We recommend fixing the issues reported here.

- **There is duplicate code between `LegionFixedPriceSale.sol` and `LegionSealedBidAuction.sol`.** Most of the code in these two files is identical. Even though this is not a security issue right now, it can lead to situations in which a bug is fixed in one of the contracts but not in the other, or a code change is made to only one of the contracts. Extract all common code into a parent contract or otherwise reduce the code duplication.
- **Internal functions receive state variables as parameters.** State variables are accessible to all functions inside a given contract by default, so they should not be used as parameters for internal functions. Instead, variables should be read directly by called functions. This is particularly an issue in `LegionFixedPriceSale.sol` and `LegionSealedBidAuction.sol`, making the code more difficult to read and understand.
- **Some state variables can be grouped into configuration structs.** Even though there is no issue in having too many state variables in a contract, it is difficult to read and follow code in which most of the variables are global. Legion has defined structs for specifying configuration parameters when sales and auctions are created, so these structs can be reused to group state variables and make the code easier to read.

G. Fix Review Results

When undertaking a fix review, Trail of Bits reviews the fixes implemented for issues identified in the original report. This work involves a review of specific areas of the source code and system configuration, not comprehensive analysis of the system.

On July 19, 2024, Trail of Bits reviewed the fixes and mitigations implemented by the Legion team for the issues identified in this report. We reviewed each fix to determine its effectiveness in resolving the associated issue.

The team made several changes to the codebase, addressing some of the Code Quality recommendations and fixing some of the reported issues. The reported PR #233 implements all fixes for the reported issues, but there were additional changes to the codebase in PR #201, consisting of the addition of an emergency withdrawal method required by regulations, and PR #222, implementing the refactor of the sale and bid contracts into a common parent contract, and the removal of state variable parameters.

Since PR #201 is not related to any of the issues in this report, it was not part of the fix review. PR #222 was partially reviewed as part of the remediation of some of the reported issues.

On August 12, 2024, PR #419 was reviewed, as it was meant to fix the remaining issues in TOB-LGN-7 and TOB-LGN-10. These fixes consisted of additional changes to the base sale, fixed price sale and sealed bid contracts.

In summary, of the 12 issues described in this report, Legion has resolved five issues, and has not resolved the remaining seven issues. For additional information, please see the Detailed Fix Review Results below.

ID	Title	Status
1	Lack of domain separation between leaves and inner nodes in Merkle Tree implementation	Unresolved
2	The decimals() function is optional in the ERC20 standard	Unresolved
3	Minimum pledge amount can be expensive if WETH token is used	Resolved
4	publishExcessCapitalResults doesn't block further refunds	Unresolved

5	Projects can withdraw capital without providing tokens if chains are different	**Unresolved**
6	Interval checks have ambiguous values	**Resolved**
7	The Project can front run the sale result publication transaction and cancel the sale	**Resolved**
8	Any investor can outbid any specific bid	**Resolved**
9	KYC checks are performed to claim tokens, not to pledge capital	**Unresolved**
10	No length validation for the sale stages	**Resolved**
11	Some token contracts implement block lists for users	**Unresolved**
12	isPrefund flag does not differentiate between amounts pledged in prefund and normal sale stages	**Unresolved**

Detailed Fix Review Results

TOB-LGN-1: Lack of domain separation between leaves and inner nodes in Merkle Tree implementation
Unresolved. The client provided the following context for this finding's fix status:

> *Acknowledged. At this stage, we do not plan to use 64-byte leaf values.*

TOB-LGN-2: The decimals() function is optional in the ERC20 standard
Unresolved. The client provided the following context for this finding's fix status:

> *Acknowledged. At this stage, we do not intend to support tokens that do not implement the optional decimals() method as defined in the ERC20 standard. If we decide to use such tokens as bidToken in the future, we will implement the recommended actions to accommodate them.*

TOB-LGN-3: Minimum pledge amount can be expensive if WETH token is used
Resolved in commit 99165db. The minimum pledge amount is now part of the sale configuration, and can be set when the sale is deployed. Configuration structures were updated accordingly.

TOB-LGN-4: publishExcessCapitalResults doesn't block further refunds
Unresolved. The client provided the following context for this finding's fix status:

> *Acknowledged. The flow is intentionally designed as follows:*
>
> 1. *In situations where a project can raise 0 (due to all participants receiving refunds), this is mandated by the MiCA regulation and cannot be bypassed. Therefore, this design is intentional.*
> 2. *The Merkle proof for sale results will always be published after the refund period has ended, ensuring all necessary information is available following excess claims or refunds. This approach guarantees that users will always be able to withdraw funds without any shortfall.*

TOB-LGN-5: Projects can withdraw capital without providing tokens if chains are different
Unresolved. The client provided the following context for this finding's fix status:

> *Acknowledged. This is a known issue, and our approach to manage this is by having the backend publish the sale results on secondary chains only after the tokens have been supplied on the main chain, where token distribution occurs.*

TOB-LGN-6: Interval checks have ambiguous values
Resolved in commit `0ebecdd`. As part of the refactoring of the sale and bid contracts, the sale end and refund period timestamps are now checked in the `LegionBaseSale` contract,

and the functions now read the state directly from the contract's state variables. Additionally, the comparisons are now correct, removing the ambiguity mentioned in the issue.

TOB-LGN-7: The Project can front run the sale result publication transaction and cancel the sale
Partially resolved in commit `5ffe7fd`. The private key reveal step is now a separate function that will require sending a second transaction to the contract after the sales results are published. This requires the investors to trust the Legion team to actually send this second transaction, as there is no real incentive for them to send it—it costs gas, and the team already has the information.

However, we still see another issue with this approach. Even though the actual individual bid amounts cannot be determined (addressing the second exploit scenario mentioned in the issue), the team can still see the transaction in the mempool, with the total amount of tokens to be allocated, and the capital raised. If again, for some reason, the Project does not like the result, they still can front run the transaction and cancel the sale.

We agree that even if the bid results are not committed in the same transaction, the project is still free to decide not to provide the tokens afterwards and leave the auction to expire. However, since the team plans to implement the two-step reveal process (as shown by the proposed fix), it is recommended that the first step "locks" the auction from being canceled, and the second step provides all of the auction results together. This way, the Project cannot front run the transaction based on the results, and the investors are assured that the private key will be published along with the rest of the information.

As part of PR #419, commit `e0a5924` further addresses this issue. The two-step reveal process now consists of a first transaction that sets a `cancelLock` flag to prevent the project from cancelling, and a second transaction that publishes all the results together, including the private key used. The test suite was modified to adapt to the new changes, and new tests were added.

TOB-LGN-8: Any investor can outbid any specific bid
Resolved in commit `811fdaf`. The salt is now checked to be equal to the sender's address, eliminating the possibility of replay attacks from different investors.

However, given the particularities of this issue and the previous one related to encryption schemes, key reveal mechanisms and public information, we still recommend auditing these specific items by a team of cryptography experts.

TOB-LGN-9: KYC checks are performed to claim tokens, not to pledge capital
Unresolved. The client provided the following context for this finding's fix status:

> *Acknowledged. This is a known issue for us, but we decided to proceed in this manner due to the reduced maintenance required for the KYC Merkle proof. In the rare instance where an unidentified account commits capital to a sale, they have two options to retrieve their funds if they choose not to proceed with KYC:*
>
> 1. *If the refund period for the sale is still open, they can use the requestRefund method to recover their funds.*
> 2. *If the refund period for the sale has closed, they can use the claimExcessCapital method to retrieve their funds.*

TOB-LGN-10: No length validation for the sale stages
Partially resolved in commit `1667625`. The commit introduces a check for maximum length values for each of the stages, for both the sales and bids. These limits are hard coded as constants in seconds: 14 days, 90 days, and 182.5 days for two weeks, three months, and six months, respectively. The invalid period error was renamed, and new test cases were introduced for invalid period configurations.

However, the minimum length for each of these stages is still not enforced; as a result, it is possible to generate a sale or bid whose stages are too short (e.g., below a block time) and are impossible to interact with.

As part of PR #419, commit `70e303a` further addresses this issue. In the configuration validation functions for the fixed price sale and the sealed bid auction, new checks were added to enforce a minimum length of one hour (3600 seconds) for each stage, in addition to the pre-existing maximum length checks. The test suite was also modified to adapt to the changes, and new tests were added for the new validation checks.

TOB-LGN-11: Some token contracts implement block lists for users
Unresolved. The client provided the following context for this finding's fix status:

> *Acknowledged. We will prepare and provide a separate document detailing how this issue will be addressed.*

TOB-LGN-12: isPrefund flag does not differentiate between amounts pledged in prefund and normal sale stages
Unresolved. The client provided the following context for this finding's fix status:

> *Acknowledged. We will maintain the current structure as it is unnecessary to store this information on-chain. The event emission provides sufficient details for backend processing and verification of the bid's status.*

H. Fix Review Status Categories

The following table describes the statuses used to indicate whether an issue has been sufficiently addressed.

Fix Status	
Status	**Description**
Undetermined	The status of the issue was not determined during this engagement.
Unresolved	The issue persists and has not been resolved.
Partially Resolved	The issue persists but has been partially resolved.
Resolved	The issue has been sufficiently resolved.