

Security Audit

# SGN and SGR Smart Contracts

---

KANSO LABS

August 2020

**VERSION 2.3**

**PRIVATE & CONFIDENTIAL**

# Table of Contents

<b>Summary</b>	<b>3</b>
<b>Disclaimer</b>	<b>4</b>
<b>Testing</b>	<b>5</b>
Analysis	5
Code Coverage	5
<b>Findings and Recommendations</b>	<b>7</b>
Mechanism	8
Stale Oracle Rates	8
SGAToSGRInitializer Initialization Sanity Checks	8
MonetaryModelState and SGRTOKEN Initialization Sanity Checks	9
ApprovalVoting Duplicated Description	9
Testing	10
Avoid after()/before() Hooks	10
Avoid Test Execution Order Assumptions	10
Missing Event Tests	11
Use OpenZeppelin's expectRevert Test Helper for Error Handling	12
Use OpenZeppelin's expectEvent Test Helper for Event Testing	12
Use OpenZeppelin's Constants Test Helper	13
Use CI	13
Time Sensitive Tests	13
Code Duplication	14
Require Node Modules at the Beginning of the File	15
Documentation	16
Off-chain Voting Mechanism	16
SECURITY.md	16
Maintainer Security Advisory	16
Miscellaneous	17
Solidity Development Environment	17
Consider Using Newer Solidity Compiler	17
Consider Using Newer Truffle Suite	17
Consider Speeding up Tests Using an In-process Ganache Provider	17
Consider Improving Solidity Coverage Integration by Using it as a Truffle Plugin	18
Artifacts Management	18
Move All Dependencies to Development Dependencies	19
Add .gitattributes for Solidity Syntax Highlighting	19
Gas Optimizations	20

Extract Test/Client-only Functions to Test Helper Contracts	20
Project and Structure	21
Add SPDX License Identifier to Contracts	21
Add SECURITY.md	21
Add README.md Badges	21
Style and Refactoring	23
Verify ERC20 Transfer Functions	23
Use Assertions for Verifying Conditions Which Should Never Be Possible	23
Prefer Revert() Over Require(false)	23
Import Libraries Before State Variables	24
Define One Contract Per Solidity File	24
Use Lowercase Variable Names	24
Inconsistent Curly Braces	25
Use Postfix Exponents	25
Index OracleRateApprover Events	26
Redundant Getters	26
Declare Visibility Modifier Before Any Custom Modifiers	27
Return Multiple Values by Name	28
<b>Appendix A: Audited Files</b>	<b>29</b>
Version 2.3	29
<b>Appendix B: Natural Logarithm Tests</b>	<b>34</b>
<b>Appendix C: Exponential Function Tests</b>	<b>40</b>
<b>Appendix D: SECURITY.md Template</b>	<b>43</b>
Security Policy	43
Reporting a Vulnerability	43
Responsible Investigation and Reporting	43
Bug Bounty Program	44
Plaintext PGP Key	44
<b>Appendix E: Setting up Github Security Policy</b>	<b>45</b>
<b>Appendix F: Setting up Maintainer Security Advisory</b>	<b>46</b>



## Summary

Kanso Labs was engaged to perform a time-boxed security review of the Sogur Core's smart contracts. The report focused on security and functional aspects of the Solidity implementation of the smart contracts. General recommendations and informational comments are also provided.

The code is excellent, very straightforward, and excellently tested.

**The team has followed most of our recommendations and updated the contracts. The issues that weren't fixed do not affect the models' safety or correctness and their implementations.**

This report is a follow-up on the previous "SGN and SGA Smart Contracts Security Audit" v1.0, v1.1, v1.2, v2.0, v2.1, and v2.2 reports, focusing on the new SGA to SGR migration and voting smart contracts, the new oracle rate approver, as well as monetary model changes. **Please see the previous reports for a complete list of recommendations.**

**No new critical security issues were identified, and the latest version of the Sogur contracts conforms to the same security standards and administrator/operator risks as the previously deployed version.**

Please see [Appendix A: Audited Files](#) for a detailed list of the audited files and their versions.



## Disclaimer

This report reflects our current understanding of known security patterns as they relate to the Sogur Core contracts. It is not an endorsement of the contracts' reliability or effectiveness, merely an assessment of their logic and robustness. KANSO Labs has not reviewed the related Sogur Core project, its back-end administration and operation system, deployment scripts, and operational or organizational security.

This report should not be viewed as investment or legal advice. It is provided by KANSO Labs as is, without warranty of any kind, express or implied. In no event shall KANSO Labs be liable concerning any subject matter relating to this report, including any decisions based upon it.

Formal security audits are not enough to guarantee secure smart contracts. KANSO Labs recommends that Sogur Core establish a bug bounty program, setting a period during which security researchers attempt to break the token's invariants, in turn encouraging further analysis of the contract.

Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit are recommended after the issues covered are fixed.



# Testing

## Analysis

Congratulations on writing such a thorough test suite! There are **50,078** tests in total, including an extensive test matrix, with most of them covering the integration of all contracts into the SGN, and the SGA contracts:

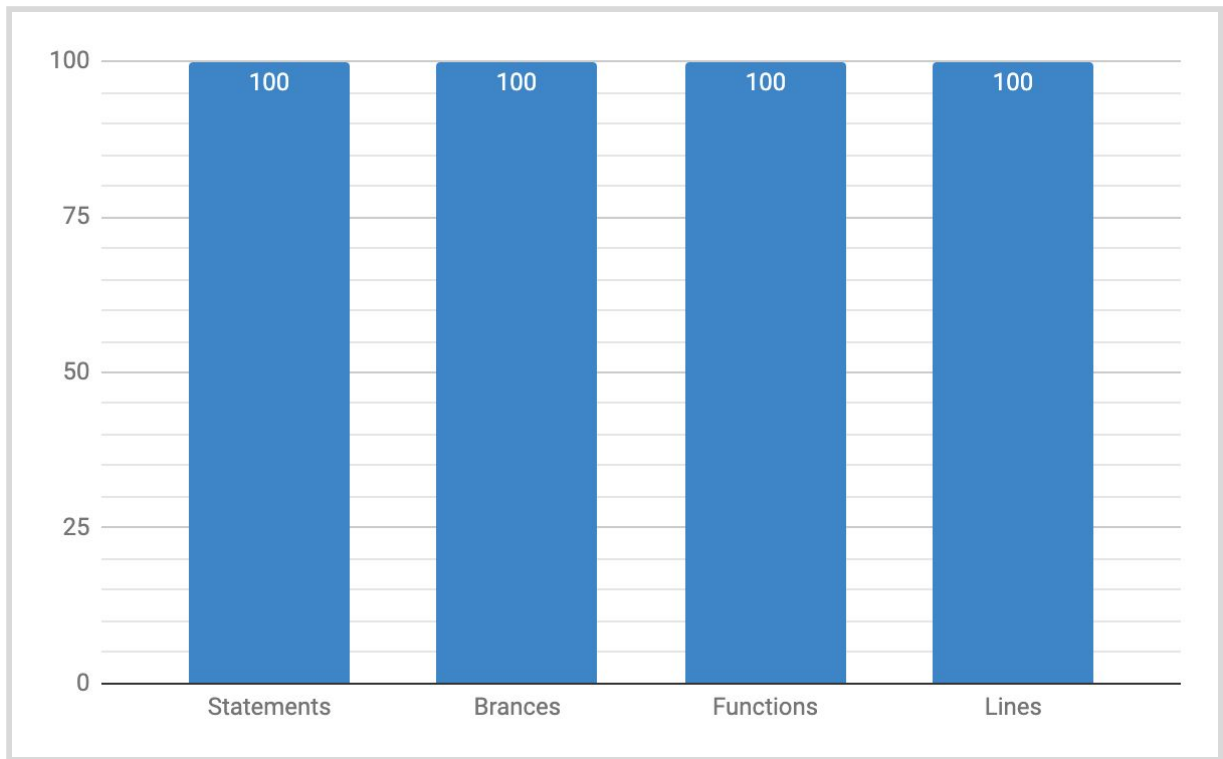
**Well done!**

## Code Coverage

The following table shows the test code coverage for this version of the Saga Core smart contracts:

File	Statements	Branches	Functions	Lines
authorization/	100%	42/42	100%	10/10
authorization/interfaces/	100%	0/0	100%	0/0
contract_address_locator/	100%	27/27	100%	16/16
contract_address_locator/interfaces/	100%	0/0	100%	0/0
migrations/	100%	33/33	100%	28/28
migrations/helpers/	100%	4/4	100%	0/0
saga-genesis/	100%	173/173	100%	24/24
saga-genesis/interfaces/	100%	0/0	100%	0/0
sogur/	100%	544/544	100%	204/204
sogur/interfaces/	100%	0/0	100%	0/0
utils/	100%	17/17	100%	10/10
voting/	100%	29/29	100%	14/14
wallet_trading_limiter/	100%	61/61	100%	24/24
wallet_trading_limiter/interfaces/	100%	0/0	100%	0/0

The total code coverage of **100%** statements, **100%** branches, **100%** functions, and **100%** lines is very high and in full compliance with the standards of the industry.





## Findings and Recommendations

Described below are the findings and recommendations for the audited version<sup>1</sup>.

Kudos to the team of their exceptional architecture, productive utilization of interfaces, and precise setting of function visibility in all of their code. The code is very modular and looks thoughtfully designed.

**No new critical security issues were identified, and the latest version of the Sogur contracts conforms to the same security standards and administrator/operator risks as the previously deployed version.**

---

<sup>1</sup> Please see [Appendix A: Audited Files](#) for a detailed list of the audited files and their versions



## Mechanism

### Stale Oracle Rates

In the current version, the **OracleRateApprover** uses **Chainlink's** latest answer to verify whether higher/lower rates are approved, but it doesn't take into account that a long period of time might have passed between an oracle update. Hence a stale rate will be used:

```
163     function getOracleLatestRate() internal view returns (uint256) {  
164         int256 latestAnswer = oracleRateAggregator.latestAnswer();  
165         assert(latestAnswer > 0);  
166         return uint256(latestAnswer);  
167     }
```

We recommend retrieving the oracle update time (via **Chainlink's** **latestTimestamp** method) and performing a sanity check on outdated rates.

**UPDATE:** The team has acknowledged the issue and will address it in the future.

*The suggested change will be implemented in future versions of code.*

### SGAToSGRInitializer Initialization Sanity Checks

We can see that the **SGAToSGRInitializer** contract a **RedButton** contract's address and make sure that it's enabled, before performing the initialization, but it doesn't check whether this is the same **RedButton** instance which is being used by the previously deployed **SGATokenManager** (and therefore **SGAToken**) contract.

We recommend removing the **\_redButtonAddress** constructor parameter and deriving the **RedButton** address from the **\_sgaTokenAddress** parameter instead.

**UPDATE:** Since it's a one-time migratory code, the team has decided not to address it at this time.

*We understand the logic behind the suggested change but given that this part of the code is going to be run only once and by us in a controlled manner, we believe there are no implications to keeping the code as it is and prefer to do so.*

## MonetaryModelState and SGRToken Initialization Sanity Checks

Even though we didn't identify any existing issues with the current SGA to SGR initialization parameters, we would like to recommend additionally verifying them on the receiving contracts side as well.

In `contracts/sogur/MonetaryModelState.sol`:

```
41  */
42  function init(uint256 _sdrTotal, uint256 _sgrTotal) external onlyIfNotInitialized only(_SGAToSGRInitializer_) {
43      initialized = true;
44      sdrTotal = _sdrTotal;
45      sgrTotal = _sgrTotal;
46      emit MonetaryModelStateInitialized(msg.sender, _sdrTotal, _sgrTotal);
47  }
48
```

In `contracts/sogur/SGRToken.sol`:

```
111  */
112  * @param _sgaToSGRTokenExchangeAddress the contract address.
113  * @param _sgaToSGRTokenExchangeSGRSupply SGR supply for the SGAToSGRTokenExchange contract.
114  */
115  function init(address _sgaToSGRTokenExchangeAddress, uint256 _sgaToSGRTokenExchangeSGRSupply) external onlyIfNotInit
116      require(_sgaToSGRTokenExchangeAddress != address(0), "SGA to SGR token exchange address is illegal");
117      initialized = true;
118      _mint(_sgaToSGRTokenExchangeAddress, _sgaToSGRTokenExchangeSGRSupply);
119      emit SGRTokenInitialized(msg.sender, _sgaToSGRTokenExchangeAddress, _sgaToSGRTokenExchangeSGRSupply);
120  }
```

**UPDATE:** Since it's a one-time migratory code, the team has decided not to address it at this time.

*We understand the logic behind the suggested change but given that this part of the code is going to be run only once and by us in a controlled manner, we believe there are no implications to keeping the code as it is and prefer to do so.*

## ApprovalVoting Duplicated Description

We can see that it's possible to deploy multiple instances of the **ApprovalVoting** contract having exactly the same **description**, **startBlock**, and **endBlock** parameters.

We recommend to either document how this situation will be mitigated off-chain or to consider implementing a global **ApprovalVoting** registry and check for uniqueness during the registration of the proposal.

**UPDATE:** The team has acknowledged the issue and will address it in the future.

*The current voting contract is a stand-alone one. future versions of the voting platform will support multiple votes and then the suggested change would become relevant.*

## Testing

### Avoid after()/before() Hooks

Since **after()/before()** hooks affect all the test cases in its scope, they can cause test cases to unnecessarily share/reuse the state.

For example, in **test/utils/AdminableUnitTest.js** the **Adminable** instance **hAdminable** is created only once and shared between all the test cases:

```
11 before(async function() {  
12     hAdminable = await artifacts.require("Adminable").new();  
13 });
```

This creates an undesired state sharing and co-dependency between test cases and can lead to unwanted results and reduced test quality.

We recommend making sure that every test case is using a pristine **hAdminable** instance every time by:

1. Instantiating **hAdminable** in a **beforeEach()** hook instead.
2. Setting the **hAdminable** variable to **const**.
3. Set up any state prerequisites (e.g., an **Adminable** with few pre-existing admins) in the context of the test case itself (preferably, using the **context()** alias for readability).

**UPDATE:** The issue hasn't been resolved yet.

### Avoid Test Execution Order Assumptions

Don't assume **it()** specific order of execution as it prevents any contributor from running the suites in a randomized order nor allows single/subset test debugging (e.g., using the **only** directive).

For example, in **test/authorization/AuthorizationDataSourceUnitTest.js**, the second test case assumes the correct execution of the first test case. Thus, it is obliged to use **SEQUENCE\_NUM + 1** instead of **SEQUENCE\_NUM**:

```

60 describe("functionality assertion:", function() {
61     it("function upsertOne should insert the wallet", async function() {
62         let response = await hAuthorizationDataSource.upsertOne(wallet, SEQUENCE_NUM, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
63         await verify(response.logs[0], "WalletSaved", wallet, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
64         assert((await hAuthorizationDataSource.walletCount()).equals(1));
65     });
66     it("function upsertOne should update the wallet", async function() {
67         let response = await hAuthorizationDataSource.upsertOne(wallet, SEQUENCE_NUM + 1, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
68         await verify(response.logs[0], "WalletSaved", wallet, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
69         assert((await hAuthorizationDataSource.walletCount()).equals(1));
70     });
71     it("function upsertOne should not update the wallet", async function() {
72         let response = await hAuthorizationDataSource.upsertOne(wallet, SEQUENCE_NUM + 1, !IS_AUTHORIZED, !IS_RESTRICTED, TRADING_CLASS);
73         await verify(response.logs[0], "WalletNotSaved", wallet, IS_AUTHORIZED, IS_RESTRICTED, TRADING_CLASS);
74         assert((await hAuthorizationDataSource.walletCount()).equals(1));
75     });

```

**UPDATE:** The issue hasn't been resolved yet.

## Missing Event Tests

The following events aren't being tested in the test suite:

- In **contracts/utills/Adminable.sol**:
  - AdminAccepted
  - AdminRejected
- In **contracts/sogur/SGATokenManager.sol**:
  - DepositCompleted
  - ExchangeEthForSgaCompleted
  - ExchangeSgaForEthCompleted
  - MintSgaForSgnHoldersCompleted
  - TransferEthToSgaHolderCompleted
  - TransferSgaToSgnHolderCompleted
  - WithdrawCompleted
- In **contracts/sogur/MonetaryModel.sol**:
  - MonetaryModelBuyCompleted
  - MonetaryModelSellCompleted
- In **contracts/sogur/PaymentManager.sol**:
  - PaymentRegistered
  - PaymentSettled

We recommend testing every emitted event and its parameters.

**UPDATE:** The team has resolved some of the missing tests and will handle the rest in the future:

Contract Name	Event Name	Status
Adminable	AdminAccepted	Fixed

Adminable	AdminRejected	Fixed
SGNTokenManager	ExchangeSgnForSgaCompleted	Fixed
SGNTokenManager	MintSgnForFoundationCompleted	Fixed
SGATokenManager	ExchangeEthForSgaCompleted	Fixed
SGATokenManager	ExchangeSgaForEthCompleted	Fixed
SGATokenManager	MintSgaForSgnHoldersCompleted	Fixed
SGATokenManager	TransferEthToSgaHolderCompleted	Fixed
SGATokenManager	TransferSgaToSgnHolderCompleted	Fixed
SGATokenManager	DepositCompleted	Fixed
SGATokenManager	WithdrawCompleted	Fixed
MonetaryModel	MonetaryModelBuyCompleted	Fixed
MonetaryModel	MonetaryModelSellCompleted	Fixed
PaymentManager	PaymentRegistered	-
PaymentManager	PaymentSettled	Fixed

## Use OpenZeppelin's expectRevert Test Helper for Error Handling

Consider using [@openzeppelin/test-helpers](https://github.com/OpenZeppelin/openzeppelin-test-helpers)'s **expectRevert** for error handling.

In addition to avoiding implementing it yourself in **helpers/Utils.js** as **catchRevert**, this helper supports verifying the actual error message, which is utterly important. It's of incredible value to confirm that a transaction has failed with a specific expected error.

For example, if you were expecting a transaction to fail with the "ERR\_INVALID\_SUPPLY" error message, but accidentally passed an incorrectly encoded parameter and caused the VM to abort. Using **catchRevert**, you won't be able to distinguish between these cases and assume that your test passes as expected, while it should have failed.

## Use OpenZeppelin's expectEvent Test Helper for Event Testing

Consider using [@openzeppelin/test-helpers](https://github.com/OpenZeppelin/openzeppelin-test-helpers)'s **expectEvent** for event testing. This test helper allows a very convenient and clean way to test for emitted events, their named parameters, their non-existence, emitting from constructors, etc.

## Use OpenZeppelin's Constants Test Helper

Consider using [@openzeppelin/test-helpers](#)'s **constants** when needing an invalid **0x0** address, instead of redefining it in multiple places by yourself (as `utils.zeroAddress`, `"0x0"`, `"0x00"`, etc).

## Use CI

Consider integrating publicly available automatic CI to the project (e.g., [Travis CI](#), [CircleCI](#), and so forth), including both invocations of the test suite and document its code coverage (e.g., [Codecov](#)).

**UPDATE:** The issue wasn't resolved yet:

*Not yet handled (this is a DevOps issue).*

## Time Sensitive Tests

We recommend verifying time-sensitive/aware tests using time manipulation capabilities, provided by **ganache** (for example, using [OpenZeppelin's time.js helper](#)), instead of state manipulating mockups.

For example, in `test/sogur/MintManagerUnitTest.js`:

```
15
16 describe("functionality assertion:", function() {
17   before(async function() {
18     await hContractAddressLocatorProxy.set("IDataSource" , hDataSource .address);
19     await hContractAddressLocatorProxy.set("ITimeManager" , hTimeManager .address);
20     await hContractAddressLocatorProxy.set("IMintListener", hMintListener.address);
21   });
22   for (let expired of [false, true]) {
23     for (let index = 0; index < 10; index++) {
24       it(`expired = ${expired}, index = ${index}`, async function() {
25         await hTimeManager.setExpired(expired);
26         await hMintManager.setIndex(index);
27         let state = await hMintManager.isMintingStateOutdated();
28         await hMintManager.updateMintingState();
29         let expected = state ? index + 1 : index;
30         let actual = await hMintManager.getIndex();
31         assert(actual.equals(expected), `expected = ${expected}, actual = ${actual}`);
32       });
33     }
34   }
35 });
36
37
```

Instead of allowing the interval to expire naturally (e.g., by manipulating the time using **time.js**'s **increase** or **increaseTo**) methods, the test forces the internal to expire using the **TimeManagerMockup** in `contracts/sogur/helpers/MintingPointTimersManagerMockup.sol`:



```

31     function setExpired(bool _state) external {
32         isExpired = _state;
33     }

```

Besides, the **setExpired** method expires **all** the intervals (similarly to **setRunning** setting **all** the intervals to running), thus dangerously affecting the state globally.

**UPDATE:** The team has reassured that time manipulation via mockups is only used in unrelated unit-tests:

*The only contract which depends on time is `TimeManager`. And indeed, in this contract's unit-test, we manipulate the time via `evm\_increaseTime`.*

*In all other unit-tests, since we use the `TimeManagerMockup` contract instead, we do not need to manipulate the time in this manner.*

## Code Duplication

There appears to be some code duplication and repetition in tests.

For example, between **test/sogur/PriceBandCalculatorExtUnitTest.js** and **test/sogur/ModelCalculatorExtUnitTest.js**:

<pre> 25 26     describe("accuracy assertion:", function() { 27         for (let func of [buy, sell]) { 28             for (let row = 0; row &lt; intervallists.length; row++) 29                 for (let col = 0; col &lt; intervallists[row].length; col++) 30                     let [minN, maxN, minR, maxR, alpha, beta] = 31                         let incN = maxN.minus(minN).dividedBy(NUM_OF_TESTS_PER_INTERVAL); 32                     for (let i = 0; i &lt; NUM_OF_TESTS_PER_INTERVAL; i++) 33                         let sqaTotal = minN.plus(incN.times(i)).truncated(); </pre>	<pre> 17 18     describe("accuracy assertion:", function() { 19         for (let row = 0; row &lt; intervallists.length; row++) 20             for (let col = 0; col &lt; intervallists[row].length; col++) 21                 let [minN, maxN, minR, maxR, alpha, beta] = 22                     let incR = maxR.minus(minR).dividedBy(NUM_OF_TESTS_PER_INTERVAL); 23                 for (let i = 0; i &lt; NUM_OF_TESTS_PER_INTERVAL; i++) 24                     let newR = minR.plus(incR.times(i)).truncated(); 25                 it(`interval \${row} \${col} - test \${i} -`, function() { </pre>
---	--

For another example, between **test/saga-genesis/SGNAuthorizationManagerUnitTest.js** and **test/sogur/SGAAuthorizationManagerUnitTest.js**:

<pre> 13     before(async function() { 14         await hContractAddressLocatorProxy.set("IAuthorizationManager", hContractAddressLocatorProxy.get("IAuthorizationManager")); 15     }); 16     test(isAuthorizedToSell, accounts.slice(0,1)); 17     test(isAuthorizedToTransfer, accounts.slice(0,2)); 18     test(isAuthorizedToTransferFrom, accounts.slice(0,3)); 19 }); 20 21 function test(func, wallets) { 22     for (let i = 0; i &lt; 4 * wallets.length; i++) { 23         it(`function \${func.name}, combination \${i}`, function() { 24             let expected = true; 25             for (let j = 0; j &lt; wallets.length; j++) { 26                 authorized = ((Math.floor(i / 4 * j) &gt; 0) ? true : false); 27                 restricted = ((Math.floor(i / 4 * j) &gt; 0) ? true : false); 28                 expected = expected &amp;&amp; authorized &amp;&amp; !restricted; 29             } 30             await hAuthorizationDataSource.set(wallets[i], {authorized: expected, restricted: restricted}); </pre>	<pre> 13     before(async function() { 14         await hContractAddressLocatorProxy.set("IAuthorizationManager", hContractAddressLocatorProxy.get("IAuthorizationManager")); 15     }); 16     test(isAuthorizedToBuy, accounts.slice(0,1)); 17     test(isAuthorizedToSell, accounts.slice(0,1)); 18     test(isAuthorizedToTransfer, accounts.slice(0,2)); 19     test(isAuthorizedToTransferFrom, accounts.slice(0,3)); 20 }); 21 22 function test(func, wallets) { 23     for (let i = 0; i &lt; 4 * wallets.length; i++) { 24         it(`function \${func.name}, combination \${i}`, function() { 25             let expected = true; 26             for (let j = 0; j &lt; wallets.length; j++) { 27                 authorized = ((Math.floor(i / 4 * j) &gt; 0) ? true : false); 28                 restricted = ((Math.floor(i / 4 * j) &gt; 0) ? true : false); 29                 expected = expected &amp;&amp; authorized; 30             } 31             await hAuthorizationDataSource.set(wallets[i], {authorized: expected, restricted: restricted}); </pre>
--	--

For another example, in **test/sogur/SGATokenUnitTest.js**:

```

46   let tokenContractSgaBalance = await hSGAToken.b
47   let sourceWalletSgaBalance = await hSGAToken.b
48   await sendTransaction(sourceWallet, AMOUNT);
49   assert(tokenContractEthBalance.plus(AMOUNT).equ
50   assert(tokenContractSgaBalance.plus(AMOUNT).equ
51   assert(sourceWalletSgaBalance .plus(AMOUNT).equ
52   });
53   it("function exchange should be valid in terms of b
54   let tokenContractEthBalance = await web3.eth.ge
55   let tokenContractSgaBalance = await hSGAToken.t
56   let sourceWalletSgaBalance = await hSGAToken.b
57   await hSGAToken.exchange({from: sourceWallet, v
58   assert(tokenContractEthBalance.plus(AMOUNT).equ
59   assert(tokenContractSgaBalance.plus(AMOUNT).equ
60   assert(sourceWalletSgaBalance .plus(AMOUNT).equ
61   });
62   it("function transfer should be valid in terms of b
63   let sourceWalletSgaBalance = await hSGAToken.ba

```

We recommend reusing shared logic in tests (using a shared function, utilizing [Mocha Shared Behaviours](#), etc.).

**UPDATE:** The team hasn't resolved these issues yet since it's a low priority.

## Require Node Modules at the Beginning of the File

We recommend avoiding requiring node modules in the middle of the test files to prevent any module loading side effects during the tests. It is also better to use it at the beginning of your file to avoid blocking calls while your tests run.

For example, in **test/sogur/PaymentManagerUnitTest.js**:

```

44   let catchRevert = require("../exceptions.js").catchRevert;
45

```

**UPDATE:** The team hasn't resolved the issue yet.





## Documentation

### Off-chain Voting Mechanism

Please consider adding documentation of how the off-chain voting mechanism will work (e.g., Sybil attack resistant snapshotting mechanism, etc.).

### SECURITY.md

We highly recommend adding SECURITY.md file to every repository and standardizing the way security incidents are **responsibly** disclosed.

SECURITY.md should describe:

1. How to report security incidents and vulnerabilities.
2. How to report anonymously (e.g., not to be blamed if the vulnerability is being exploited).
3. How to make sure that the report is being delivered to the right personnel (e.g., by providing an email as well as a PGP key).
4. Information regarding any bounty programs and rewards.

Please see [Appendix D: SECURITY.md Template](#) for an example.

We also recommend adding your SECURITY.md using Github's new Security Policy feature. Please see [Appendix E: Setting Github Security Policy](#) for more information.

### Maintainer Security Advisory

We highly recommend using Github's new Maintainer Security Advisory feature (which is currently in beta but works like a charm nonetheless). Now maintainers can have a private workspace to discuss, fix, and publish security advisories to people who rely on their projects right within GitHub (without tipping off would-be hackers).

Please see [Appendix F: Setting up Maintainer Security Advisory](#) for more information.

## Miscellaneous

### Solidity Development Environment

#### Consider Using Newer Solidity Compiler

All the system contracts are currently compiled using Solidity v0.4.25. We highly recommend upgrading to Solidity [v0.5.17](#) or [v0.6.12](#), taking advantage of the new and safer language features, improved ABI encoder, improved optimizer, and much more.

This will also allow you to take advantage of newer community SDKs (such as [@openzeppelin/contracts v2.5.1](#)).

#### Consider Using Newer Truffle Suite

Currently, the project utilizes **truffle** v4.1.16. Please consider upgrading to **truffle** [5.1.41](#), to take advantage of much more improved development and debugging experience, including faster ganache provider and integrated code coverage (please see below).

#### Consider Speeding up Tests Using an In-process Ganache Provider

Following up with the recommendation to upgrade **truffle** to a newer version, you can consider switching from **ganache-cli** to an in-process **ganache-core** provider.

For example, like so:

```
32 const ganache = require('ganache-core');
33
34 module.exports = {
35   networks: {
36     development: {
37       network_id: '*',
38       provider: ganache.provider({
39         total_accounts: 30,
40         defaultEtherBalance: 1000,
41       }),
42     },
43   },
44 }
```

This will also allow you to remove **ganache-cli** from project dependencies.

Similarly, you can use [@openzeppelin/test-environment](#), which will require slightly more refactoring but will achieve similar performance benefits.

## Consider Improving Solidity Coverage Integration by Using it as a Truffle Plugin

Following up with the recommendation to upgrade **truffle** to a newer version, you can consider switching from externally managed **solidity-coverage** to a **truffle** plug-in, like so:

```
34 module.exports = {
35   networks: {
36     development: {
37       network_id: '*',
38       provider: ganache.provider({
39         total_accounts: 30,
40         defaultEtherBalance: 1000,
41       }),
42     },
43   },
44   plugins: ['solidity-coverage'],
45   compilers: {
46     solc: {
```

This will also allow you to remove **solidity-coverage** from project dependencies and run it by executing “**truffle run coverage**”.

## Artifacts Management

In the current project setup, auto-generated Solidity artifacts (i.e., abi, bin, and JSON files) are committed to the source control. This unnecessarily adds to the repository history and size as well as introduces inconsistencies between the code and resulted binaries (e.g., a test can run against a committed binary instead of against the modified code).

I recommend removing these files from the repository<sup>2</sup> and publishing them as part of the npm repository<sup>3</sup> instead:

- Publish versions to npm via “**npm publish**”.
- Add a **files** attribute to your **package.json** file, specifying which files and artifacts should/should not be included in the package. For example:

```
11   "main": "truffle-config.js",
12   "files": [
13     "solidity/build",
14     "solidity/contracts",
15     "contracts/helpers"
16   ],
17   "scripts": {
```

This way, every project which will include the package, will get easy access to its contracts (excluding tests), and current/past artifacts, while the Github repo remains pristine.

---

<sup>2</sup> You can also consider carefully removing them from the whole git history to reduce its size (like it's being done in the case of an accidentally committed secret, for example)

<sup>3</sup> This is also the approach other community packages, such as [@openzeppelin/contracts](https://github.com/openzeppelin/contracts), are taking.

## Move All Dependencies to Development Dependencies

Following up with the previous recommendations since consuming the **Sogur** package will be mostly used to get its contracts, binaries, and ABI, which don't require compilation or preprocessing - none of the current dependencies are needed to be pulled as well. In such cases, moving these dependencies to the **devDependencies** development dependencies section will allow developers to continue working on the **Sogur** while significantly reducing the package size and dependency management for the consuming packages.

As another advantage, this approach will prevent outdated (and potentially vulnerable) dependencies from infecting its consumers.

## Add .gitattributes for Solidity Syntax Highlighting

Consider adding a **.gitattributes** file to enable Solidity syntax highlighting in external browsers and tools. It should only include the following line:

```
◆ .gitattributes  
1 *.sol linguist-language=Solidity
```

## Gas Optimizations

### Extract Test/Client-only Functions to Test Helper Contracts

It'd seem that in some instances, some convenience functions were added to the main smart contracts.

A typical example is convenience getters, such as an array length getters (e.g., **ApprovalVoting**'s **getTotalVoters** function).

You can consider extracting these functions from the main smart contracts and then:

- For tests, you can add them to a wrapper/mock test contract (e.g., an **ApprovalVotingMock**, inheriting from **ApprovalVoting** and adding the **getTotalVoters** function).
- For client SDKs, you can consider getting the length by retrieving the array in pages and checking when an invalid **0x0** item is retrieved. Hopefully, querying array lengths should be officially supported by future web3.js/etherjs versions.

---

## Project and Structure

### Add SPDX License Identifier to Contracts

I recommend adding an [SPDX License Identifier](#) to each contract. Not only that, but it also simplifies licensing issues by the community<sup>4</sup>.

For example, you can see how it was added in [@openzeppelin/contracts](#) in this commit: <https://github.com/OpenZeppelin/openzeppelin-contracts/commit/56de324afea13c4649b00ca8c3a3e3535d532bd4>.

### Add SECURITY.md

I recommend adding SECURITY.md file to every repository and standardizing the way security incidents are **responsibly** disclosed.

SECURITY.md should describe:

1. How to report security incidents and vulnerabilities.
2. How to report anonymously (e.g., not to be blamed if the vulnerability is being exploited).
3. How to make sure that the report is being delivered to the right personnel (e.g., by providing an email as well as a PGP key).
4. Information regarding any bounty programs and rewards.

Please see [Appendix D: SECURITY.md Template](#) for an example.

It's preferred to add your SECURITY.md using Github's new Security Policy feature. Please see [Appendix E: Setting Github Security Policy](#) for more information.

### Add README.md Badges

Please consider adding informational badges (such as npm version, CI status, coverage status, useful links, etc.) to your README.md.


For example, please see the badges at <https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/README.md>:


---

<sup>4</sup> Please keep in mind that not having it will emit a warning on Solidity v0.6.8 and later.





📄 README.md

 **OpenZeppelin** | contracts

docs 

npm v3.0.1

circleci  passing

codecov  98%

## Style and Refactoring

### Verify ERC20 Transfer Functions

Even though the **SGAToSGRTokenExchange** contract is designed to solely work with **SGAToken** and **SGRToken**, as a general good practice, we'd recommend verifying the boolean status return of both **transferFrom** and **transfer** ERC20 functions:

```
57  */
58  */
59  function handleExchangeSGAToSGRFor(address _sgaHolder) internal {
60      uint256 allowance = sgaToken.allowance(_sgaHolder, address(this));
61      require(allowance > 0, "SGA allowance must be greater than zero");
62      uint256 balance = sgaToken.balanceOf(_sgaHolder);
63      require(balance > 0, "SGA balance must be greater than zero");
64      uint256 amountToExchange = allowance.min(balance);
65
66      sgaToken.transferFrom(_sgaHolder, SGA_TARGET_ADDRESS, amountToExchange);
67      sgrToken.transfer(_sgaHolder, amountToExchange);
68      emit ExchangeSgaForSgrCompleted(_sgaHolder, amountToExchange);
69  }
70  }
71  }
```

### Use Assertions for Verifying Conditions Which Should Never Be Possible

Prefer **assert()** over **require()** when verifying the correctness of internal state and logic to prevent anything terrible from happening, while **require()** should be used to ensure valid conditions, such as inputs, or contract state variables are met, or to validate return values from calls to external contracts.

For example, in **contracts/saga/OracleRateApprover.sol**:

```
137  function approveRate(uint256 _rateN, uint256 _rateD) internal view returns (bool) {
138      assert(_rateN > 0);
139      assert(_rateD > 0);
140      bool success = true;
141  }
```

Both **\_rateN** and **\_rateD** are being sent via an external call to either **approveHighRate** or **approveLowRate**.

### Prefer Revert() Over Require(false)

Prefer **revert(msg)** over **require(false, msg)** whenever the control flow needs to be aborted unconditionally.

For example, in **contracts/sogur/TransferOnlySGATokenManager.sol**:



```

41  */
42  function exchangeEthForSga(address _sender, uint256 _ethAmount) external returns (uint256) {
43      require(false, "SGA token has been deprecated. Use SGR token instead");
44      _sender;
45      _ethAmount;
46      return 0;
47  }
48
49  /**
50   * @dev Exchange SGA for ETH.
51   * @param _sender The address of the sender.
52   * @param _sgaAmount The amount of SGA received.
53   * @return The amount of ETH that the sender is entitled to.
54   */
55  function exchangeSgaForEth(address _sender, uint256 _sgaAmount) external returns (uint256) {
56      require(false, "SGA token has been deprecated. Use SGR token instead");
57      _sender;
58      _sgaAmount;
59      return 0;
60  }

```

## Import Libraries Before State Variables

We recommend importing libraries before defining and mutable/immutable state variables in the code.

For example, in **contracts/saga/OracleRateApprover.sol**:

```

15  contract OracleRateApprover is IRateApprover, ContractAddressLocatorHolder, Claimable {
16      string public constant VERSION = "1.0.0";
17
18      using SafeMath for uint256;
19
20      uint256 public constant MILLION = 1000000;
21      uint256 public constant ORACLE_RATE_PRECISION = 100000000;

```

## Define One Contract Per Solidity File

Please consider using a separate file for each contract/interface, as recommended by the [Solidity style guide](#).

For example, there is an additional interface in **contracts/migrations/SGAToSGRInitializer.sol**.

## Use Lowercase Variable Names

We recommend avoiding capitalized variable names, according to [Solidity v0.4.25 Style Guide](#).

For example, in **contracts/saga/OracleRateApprover.sol**:

```

142     if (!isApproveAllRates) {
143         uint256 A = (getOracleLatestRate()).mul(_rateD);
144         uint256 B = A.mul(MILLION);
145         uint256 C = A.mul(rateDeviationThreshold);
146         uint256 rate = (_rateN.mul(ORACLE_RATE_PRECISION)).mul(MILLION);
147     }

```

## Inconsistent Curly Braces

Please check **contracts/voting/ApprovalVoting.sol** for an example of inconsistent curly braces style:

```

129     /**
130     function getAllVoters() external view returns (address[] memory) {
131         return voters;
132     }
133
134     /**
135     * @dev Vote for proposal.
136     */
137     function voteFor() public
138     {
139         castVote(true);
140     }
141
142     /**
143     * @dev Vote against proposal.
144     */
145     function voteAgainst() public
146     {
147         castVote(false);
148     }
149

```

## Use Postfix Exponents

We recommend, wherever applicable, to define the powers of 10 constants using the **e** postfix exponent.

For example, in **contracts/saga-genesis/SGNConversionManager.sol**, the **DENOMINATOR** constant:

```

7     uint256 public constant DENOMINATOR = 1000000;

```

Can be defined as:

```

7     uint256 public constant DENOMINATOR = 1e6;

```

**UPDATE:** The team has decided to keep the current style:

*We prefer to use 1000000 (and not 1e6 as you suggested) because we'd like to display an easily readable set of conversion ratios (105 values between 0 and 15).*

Since the `numerators` are all denoted in full format, we believe that we may as well denote the `DENOMINATOR` in the same manner.

## Index OracleRateApprover Events

Consider indexing the `_oracleRateAggregatorAddress` and the `_oracleRateAggregatorAddress` parameters of the `OracleRateAggregatorSaved` and the `OracleRateAggregatorNotSaved` events respectively.

## Redundant Getters

In some places, for public state variables, an additional explicit getter is defined.

For example, in `contracts/sogur/MintManager.sol`:

```
9  contract MintManager is IMintManager, ContractAddressLocatorHolder {
10     uint256 public index;
```

We can later see that an explicit `getIndex` getter is defined:

```
38     function getIndex() external view returns (uint256) {
39         return index;
40     }
```

For another example, in `contracts/sogur/MonetaryModelState.sol`:

```
20     function getSdrTotal() external view returns (uint256) {
21         return sdrTotal;
22     }
23
24     function getSgaTotal() external view returns (uint256) {
25         return sgaTotal;
26     }
```

For another example, in `contracts/sogur/voting/ApprovalVoting.sol`:

```
129     /*
130     function getAllVoters() external view returns (address[] memory) {
131         return voters;
132     }
133
```

**UPDATE:** The team has decided to keep these additional getters for improved code readability and consistency:

These getters are not redundant. Because we use them also from the on-chain (and not only from the off-chain):

1. We want to use names that start with `get`, and naming a variable this way would be very misleading.
2. Solidity's implicit getters cannot be properly exposed in an `interface`, due to a technical limitation (the compiler generates `public` getters, while `interface` functions must be declared `external`).

## Declare Visibility Modifier Before Any Custom Modifiers

According to [Solidity v0.4.25 Style Guide](#), visibility modifiers for a function should come before any custom modifiers.

For example, in **contracts/sogur/SGAToken.sol**:

```
22  
23     constructor(IContractAddressLocator contractAddressLocator) ContractAddressLocatorHolder(contractAddressLocator) public {}  
24
```

The **public** visibility modifier should come before the **ContractAddressLocatorHolder** super.

For another example, also in **contracts/sogur/SGAToken.sol**:

```
31  
32     function() payable external {  
33         uint256 amount = getSGATokenManager().exchangeEthForSga(msg.sender, msg.value);  
34         _mint(msg.sender, amount);  
35     }  
36
```

The **external** visibility modifier should come before the **payable** modifier.

**UPDATE:** The team has resolved most of the issues:

*Partially done (external visibility modifier moved before payable modifier). As for the other suggestion:*

*Indeed, according to the [Solidity v0.4.25 Style Guide](#), visibility modifiers for a function should come before any custom modifiers. However:*

1. Calling the 'super' doesn't quite "qualify" as a modifier.
2. The [Arguments for Base Constructors documentation example](#) shows that the visibility modifier indeed appears after the 'super' and not before it.

## Return Multiple Values by Name

Whenever a function returns multiple values (especially with the same type), we recommend returning them by name. This will help in returning wrong values by mistake, as well as allow access to the values by their names using web3.js 1.0 and later.

For example, in **contracts/sogur/ModelDataSource.sol**:

```
72 | function getIntervalCoefs(uint256 rowNum, uint256 colNum) external view returns (uint256, uint256) {  
73 |     Interval storage interval = intervalLists[rowNum][colNum];  
74 |     return (interval.alpha, interval.beta);  
75 | }
```

Can be refactored to:

```
72 | function getIntervalCoefs(uint256 rowNum, uint256 colNum) external view returns (uint256 alpha, uint256 beta) {  
73 |     Interval storage interval = intervalLists[rowNum][colNum];  
74 |     alpha = interval.alpha;  
75 |     beta = interval.beta;  
76 | }
```

**UPDATE:** The team has decided to keep the current style:

*We decided to leave it this way because it would otherwise raise the question of why we did not apply the same in every function, which returns a single value. In addition to that, the structure of such functions (assignment into variables which are not declared inside the function, no return-statement, etc.) might be regarded as rather unclear by many developers.*

■ ■ ■

# Appendix A: Audited Files

## Version 2.3

Path	SHA256 Fingerprint
contracts/authorization/AuthorizationActionRoles.sol	0ef7fb03cc8034aadfa04add078c5df937c7bbd8dd8f6d2150836a99cae7f6ba
contracts/authorization/AuthorizationDataSource.sol	6db2611a8a06abd60991dd1f7e94745706994518d25713c579b32d8c36bc3af5
contracts/authorization/helpers/AuthorizationDataSourceMockup.sol	273ba333630b5bfc64b0ae818d41958a1a8daf4496ab5cf84ed3d89b8940a9e6
contracts/authorization/interfaces/IAuthorizationDataSource.sol	3bfadc7e2c173529c1e559ca94b33f955bf3b7201f609d7f89dcb3886f27ae6c
contracts/contract_address_locator/ContractAddressLocator.sol	217a84f2133a95d8d335a509a9edd611d8411ed2cc6b236f0dfab783cff060a
contracts/contract_address_locator/ContractAddressLocatorHolder.sol	4926991df9e76eb5503bbe55d0aa8e48c603931409d10ce11f3227b5571f9a3b
contracts/contract_address_locator/ContractAddressLocatorProxy.sol	61ae3df8fc96225e66aaf5e56f107e2e7d804f9980436ef20d5975c9efa04240
contracts/contract_address_locator/helpers/ContractAddressLocatorHolderExposure.sol	d5846c946ae7ceda0929781c4b2702b9b6a87611b746cd59d5696b111514293a
contracts/contract_address_locator/helpers/ContractAddressLocatorMockup.sol	8f3cf4d313a57dd099445daabfbc9030fd01f7b53d1dee9fe3926bd8fa9a6c96
contracts/contract_address_locator/helpers/ContractAddressLocatorProxyMockup.sol	5e9423320cad59ecc44f21e3a3f31c2f295d28296219f792db988a2f9abc8ff7
contracts/contract_address_locator/interfaces/IContractAddressLocator.sol	b10692d9dc0c6c4f79bdf3fe3b592bcaab3e481176df97b80c23b36a381e3df6
contracts/Migrations.sol	d39e214137a1664cbf3fabb8e4e28555a85399ed4285b1b30305956028ce40d1
contracts/migrations/AuthorizationDataSourceMigration.sol	446f53ce83b3a934eae57ff8a92f9a4243fc411314fd1d8ea5c88029433d6aca
contracts/migrations/helpers/SGAMonetaryModelStateMockup.sol	76ce143c744ff35044d9685ea59f7cb1933c7d5b23124fddd0f00b8103f17b84
contracts/migrations/SGAToSGRInitializer.sol	75e265cd529ba29494ea9721e6ce4a440a78fa9104f41ae93aaa1b06243b794c
contracts/migrations/SGAToSGRTokenExchange.sol	5a7f102c2422ca9de9ef4bb8ae47255ef4db251a3aeb908367c6fb816101b6e5
contracts/saga-genesis/helpers/MintManagerMockup.sol	f4dabfe591aa2fc31dfd59cb44e430c121b315eb75d519663c0bfaa1a7a3ead0
contracts/saga-genesis/helpers/SagaExchangerMockup.sol	fe75a1b481aae20aaa5a99486e4cc01bc673539fbde1548eb8a9cee9caef8eea
contracts/saga-genesis/helpers/SGNAuthorizationManagerMockup.sol	95f69ac17ef4144dde3e36050e1be7c2db50672b821ddb301cfd45ef67dff75
contracts/saga-genesis/helpers/SGNConversionManagerMockup.sol	afba6ad3302a104fd0be6ae3c2e88d77f08143038c48d2f9559d5faeb3178625
contracts/saga-genesis/helpers/SGNTokenManagerMockup.sol	bf0dc15cd6a1f7657fb270c9ed3486a20973e00e7d6278960903543b25fc6d8
contracts/saga-genesis/interfaces/IMintHandler.sol	a6a813651a7f7e99243e4e3f4062cb0489656c676632c6d868ede390b5c7b3b5
contracts/saga-genesis/interfaces/IMintManager.sol	3f98a9a4d4b9a03ab5f7853590c27820257032f2877dbb06fcc3cd18dd6e4ed1
contracts/saga-genesis/interfaces/ISagaExchanger.sol	6a3164cf75ddc4f8b660ac1b9d60da92291bca165f4be9c03ae534a9f289547b
contracts/saga-genesis/interfaces/ISGNAuthorizationManager.sol	b5d2b5a4dabb3e0afa55dac4db19711af3ed67b1d234772c15f2be3bd6d0abdf
contracts/saga-genesis/interfaces/ISGNConversionManager.sol	31e15d73e9a3a2187f7140a552d870f39364d6c56d5a38a2d4a78e9fb1bc9d48
contracts/saga-genesis/interfaces/ISGNTokenManager.sol	909d94e900bac2c37a5ee168cb59edaa0ca08cfa410f3a96024f436b4f972906
contracts/saga-genesis/interfaces/ISogurExchanger.sol	348eb4e51ac7e13310129841760f74a2e408ba51edfb2c63eeb3f237f0258afc
contracts/saga-genesis/SagaExchangerSogurAdapter.sol	2e92e4a6533cdad40552ccabcfb617bba932cb9d6d8c6ae2dc82adcce28d6eb0
contracts/saga-genesis/SGNAuthorizationManager.sol	3d1faa94c61e77bd386a7d4117d1ab6ba4290db92a845f276f208cfa68902016
contracts/saga-genesis/SGNConversionManager.sol	186dc699191bfefee077f3df3f9a4227a72fca5cb81b5e751791372317218a
contracts/saga-genesis/SGNToken.sol	be96d7f44ccc0d08a6ba63e1bb0e6d7c36fb138955f29e6a452111a828e3b4cb
contracts/saga-genesis/SGNTokenManager.sol	c451349e9fe24b8df2fc2ee8249cbb9b93080f0c2da7f7be8020388d87a4b059
contracts/saga-genesis/SGNWalletsTradingLimiter.sol	7253ac131f4119dc97718407e42a2812c062c57df7fcf39eb83a6e871a4e3f2c
contracts/sogur/BatchSetModelDataSource.sol	d5b85211b246c63fcec835336ba0bef6e7f9c73f084bb31da941a55aa70acac0

contracts/sogur/ETHConverter.sol	fe8346f9a26309a3c2afde2a59752517f07fe93f57149520bcd96e3272d2517
contracts/sogur/helpers/AggregatorInterfaceMockup.sol	646f31dd4042162b04ccd3589f90636c5866808a4621f3c019a464d60b55abe7
contracts/sogur/helpers/ERC20Mockup.sol	fac0ba3693f80fce7c079772a639023712528cbd0c07c7715b8d039686e0ab3d
contracts/sogur/helpers/ETHConverterMockup.sol	0e707e923d80bc8d790347670a58d63389cee0d122935f50ed30118caa8fc4ad
contracts/sogur/helpers/IntervallIteratorExposure.sol	e74f8a22438d0472c6d55bc82a96b7587d87e8745166420dbdf396ee57ff3d5b
contracts/sogur/helpers/IntervallIteratorMockup.sol	1734e2b7248f891cc7dd020327287ed308a33a3c050ad3e77d012997f8959a24
contracts/sogur/helpers/MintHandlerMockup.sol	a726d1cf5d24579545a50deb026299a223c6d9e67a115762a052182a6ae93650
contracts/sogur/helpers/MintingPointTimersManagerExposure.sol	1144eb52311b08e3b694ca5a07463ea7773ffaf6b4752866d0299bd80c2f1aa7
contracts/sogur/helpers/MintingPointTimersManagerMockup.sol	8a919072e1d812ef43ad73269692d2e742edf7bab36a000f4c2d71670b946acd
contracts/sogur/helpers/MintListenerMockup.sol	97dc2a944471cfff6344eb4ab2be507cb68b89af97d36abed59b1c0a22b04acc
contracts/sogur/helpers/MintManagerExposure.sol	4ba664889536c826c8600d4aa078cc9b229393ef90047073d0c9fae4e8ce9486
contracts/sogur/helpers/ModelCalculatorExposure.sol	d2e5afce939c64e2d4919919c246e1c991c9f8befdb42472cbb5a6500f5ea50f
contracts/sogur/helpers/ModelCalculatorMockup.sol	cbc16627493f426404093a0264ade36776e6edd0e7f401a89aa543a1c8b58f9e
contracts/sogur/helpers/ModelDataSourceMockup.sol	17da607582dc465b8e874c4694fecb09367c3558b13d11aceb4e1e20497955b4
contracts/sogur/helpers/MonetaryModelMockup.sol	a9a782ff82b92744906ccb222ae5db82bc21a4ae87903c5ccca581b0eb1f9328
contracts/sogur/helpers/MonetaryModelStateMockup.sol	ae94c53f4b6c58fc9e034760e9106990061b7d4c1d6181afb0536aa5ead61a0e
contracts/sogur/helpers/PaymentManagerMockup.sol	bb6bb684cacaac00c07f91e21e197816617e263071885911031f0c60d057309c
contracts/sogur/helpers/PaymentManagerUser.sol	377ea9354712fb29024e718ebaa7bddc218d401590225cf7cb10e40249908ac3
contracts/sogur/helpers/PaymentQueueExposure.sol	4551c43d6927b3bffc3b7db945b407c04d11ad270451f1a08032e1271124f06
contracts/sogur/helpers/PaymentQueueMockup.sol	4c954c7cbdc284deb28b62417e4be7f82e3ae23a2a5f3abc8c9e71aef9613d51
contracts/sogur/helpers/PriceBandCalculatorMockup.sol	867a3431afc821d02ca1887f92f789cd77c082588426452a53f44feec3449c83
contracts/sogur/helpers/RateApproverMockup.sol	382fdd299326e81a8ca172e326ae0d5ba694a2dd1fc7a3f744b80584e512d59f
contracts/sogur/helpers/ReconciliationAdjusterMockup.sol	5d9387e970606d3ccf4f7cf6950165abcf430fdf4d768a00f2e86f91a0795c16
contracts/sogur/helpers/RedButtonMockup.sol	f2e1787a70e706d8969b5e4cc5f53cad3ea81e1360595de9a0b7ad8ee8794d8c
contracts/sogur/helpers/ReserveManagerMockup.sol	6a124bba7cf94fa22606d2e514386313d54c900ac2bed72f1d21ddf014532d90
contracts/sogur/helpers/SGRAuthorizationManagerMockup.sol	4694cb764690ce6bdfd3694d9f01b6612538c0d924f46dc0b23b1272f8eff458
contracts/sogur/helpers/SGRTokenInfoMockup.sol	02b8ff74f157216e58a3b679108785dd43009c39067983592a07728fa2261f29
contracts/sogur/helpers/SGRTokenManagerMockup.sol	034d705d314239913e1f488ec42655ee16e2be2bbb07ca93bf1866c926845d54
contracts/sogur/helpers/SGRTokenMockup.sol	c4817cfff7b7db8685d6043071af006f43833dacecc59e93081b6c47e3b183ba4
contracts/sogur/helpers/TransactionLimiterMockup.sol	186e823a9550d0bd2d71c444f10c82a4d3b9aa1adbde8c2e6e7de0aaec552530
contracts/sogur/helpers/TransactionManagerMockup.sol	863e6639c161297955b272418dbab131778ae8e51747ffa4e1e4fbcdc8516e81
contracts/sogur/interfaces/IETHConverter.sol	3379863cd6fee2b60f1b66c4590d0e03faea847df0f16a3b927164d14116a4a5
contracts/sogur/interfaces/IIntervallIterator.sol	0921ce1af12f23b3616e450878e48110cf1b2a33ab24a14a031fafc1af9b5fb5
contracts/sogur/interfaces/IMintingPointTimersManager.sol	4e1cd1e3ef36dc91de85f6dc1949cb4bfbdd479cd2e5af2904f854d3b8515bcd
contracts/sogur/interfaces/IMintListener.sol	728583fd3f883d7a002c419caba67bc49e2a1c2d86a462e1d85ce05161e2b0ee
contracts/sogur/interfaces/IModelCalculator.sol	59ac15c3447126c2d53f6a539966ec3cdd04fab7512bd19c39c9a6aa560f3f8b
contracts/sogur/interfaces/IModelDataSource.sol	e6a8ac05c584ad8c9bca436d317e3de819e6cd82148a5969ce6ce838beda8f56
contracts/sogur/interfaces/IMonetaryModel.sol	63d93f45c57fb61c5fe08d8cf79a1f41c529edae2feb8f3fbcfedcc8ab27e78
contracts/sogur/interfaces/IMonetaryModelState.sol	997a84b781edc7ceca3d4d8dcd23b205d9166e5f2224092b32a2807c092e008
contracts/sogur/interfaces/IPaymentHandler.sol	d1110357b296a8be90f9bc5d0014706c4f6acb65be59116a3c4fee679f6554e0
contracts/sogur/interfaces/IPaymentManager.sol	1ea4ceae7568db63e4f3cc0da9e472c1b5be1968812d81f7a6d65fb7b11346c7
contracts/sogur/interfaces/IPaymentQueue.sol	a855f4a902f044255afc65c86cdd91e09ca53033176c04ea85524c43b52eb449
contracts/sogur/interfaces/IPriceBandCalculator.sol	b8284d97800d704cbeed3d3e6138c01cfec96b27ee1ef42b05fa0a6a9bff8261



contracts/sogur/interfaces/IRateApprover.sol	384db58185f2939dda1d5bfc3a6d516110986f96795cb7b8acaaa91eddedc4ae
contracts/sogur/interfaces/IReconciliationAdjuster.sol	00a502eded28d16a1daec89d9a581bdd827cfef6c07fac68c9ce75474d517daa
contracts/sogur/interfaces/IRedButton.sol	d0f2ed851346fd076e999d977dc9005f4e719da8aacd55951ec7d5ac9f9d7093
contracts/sogur/interfaces/IReserveManager.sol	9220b1167f3e7823836c2f80fd5e569c7c3d2ee7287d3429a0c2a0e383fb72cb
contracts/sogur/interfaces/ISGATokenManager.sol	1e9b55e2fbad433d906e6a06938cd1675734e9b628aaaae82c0fba49b8836bd2
contracts/sogur/interfaces/ISGRAuthorizationManager.sol	99cff4960fa92767e27487fa69d295f93aaf7a2e64fd09bdb53fbd07afd91ad
contracts/sogur/interfaces/ISGRTTokenInfo.sol	54603081775f2edef4e0b0003154e8a0aee31adb33149802117c4eb2f1401563
contracts/sogur/interfaces/ISGRTTokenManager.sol	2e920287c90b0060e93ddff3e484a76141f657fcd1716403cce1eb6c3dd2af2
contracts/sogur/interfaces/ITransactionLimiter.sol	45629916811f628d47bd876aadbbbe9299455d3146cb0b8e93a01b34785805a6
contracts/sogur/interfaces/ITransactionManager.sol	a2d1637d93ae3b7aa6f16f27e233b98e5bc8629e2e780be12dbba3ae49336541
contracts/sogur/Intervalliterator.sol	740255a34d83ec1588d11ef810c7bb87bfc991964d49b12801af2148888938ad
contracts/sogur/MintingPointTimersManager.sol	da32d6af04651f2b35801a2beaf7f615e41fc79415f697ebb6fc9fad8dc1c35d
contracts/sogur/MintManager.sol	f34e5ce64becae9c2b31977aaf0c8cba2d56e520219c616c2a4b921c54c4be77
contracts/sogur/ModelCalculator.sol	597a4e1c750d1cdc20c4e53fceb8c5b75bdef40dcb5464088b539a1960f38b5
contracts/sogur/ModelDataSource.sol	dd254849a388bfa19466a097a69e08de6c78d659daaf13154bf41dd9da7a2de0
contracts/sogur/MonetaryModel.sol	30a6d0e9d109100f0244c70d14663aa0fb91f89b3f6e6a7b9d72d8fa46b69d1b
contracts/sogur/MonetaryModelState.sol	ec8969f65de7b796a2f1ce97559f4d571cc8a869ccdb863080259e21ea635353
contracts/sogur/OracleRateApprover.sol	b4b53f40e6744be2563f02886e2bc6367e44ce240cfb31fb8e223adc94431688
contracts/sogur/PaymentManager.sol	fe21a50eda56ef879085d31c47f77b9da9a4c099a2689ef46d2ca575e9daf79b
contracts/sogur/PaymentQueue.sol	01328eb36ca917f36a357f18d4d896718afa76030faecd473d52abd9a3bc4
contracts/sogur/PriceBandCalculator.sol	4a50dad44c96ccc2aff37ba1c570465bd521e1fef1ce9940af98231ad3b060fd
contracts/sogur/ReconciliationAdjuster.sol	c3371be470d2809e761214ecf0c0c40f64fb2bd5e44b603459f631ca5bf5f35a
contracts/sogur/RedButton.sol	f42244ffc10cf4968db691109639294cd1af7894d1757376cb14fea326ffe467
contracts/sogur/ReserveManager.sol	5696825c1ba6a1ecc83583a87639e125517134c319d6c85a880391c054018eea
contracts/sogur/SGRAuthorizationManager.sol	65f812c6ef35498ace8dcebc404cd61e8d811c05a6158974543ac83a5f277b3f
contracts/sogur/SGRBuyWalletsTradingLimiter.sol	64df29af8bd4522d452d34b62ef3a336d7534123a07644d5f78f47ff6c7d207b
contracts/sogur/SGRSellWalletsTradingLimiter.sol	9afebac52cee17ccc1cdad69b7069d28aad4fe48ee5a2f8f014649dd24301601
contracts/sogur/SGRTToken.sol	0aca2eef497ac5c2fd1611e40472bcded3a39de1d4040f739361e5754f428efb
contracts/sogur/SGRTTokenInfo.sol	8998802ba13e295acdb9c6fea84e4c85a9449e5a538dcb01b2814a7496c50645
contracts/sogur/SGRTTokenManager.sol	8f350aab45b447e56dfd6804431ad1577f5f98a385af12d23ec2bbe0a4c44d
contracts/sogur/SGRWalletsTradingLimiter.sol	3b0c9b457e72b8c4f14dcc45d24047816fb01bdfefb3a2076b2132fc503f7429
contracts/sogur/TransactionLimiter.sol	9744e886dce98b646b45781a7185d29f02fc5368fbd5ec10766d37339d3e06cd
contracts/sogur/TransactionManager.sol	13f351fcf904b1d3098e88cf332f92a87a42cbe6d5a1e52f6bb66a78fe9bda7f
contracts/sogur/TransferOnlySGATokenManager.sol	347e80f3eb64b9c13b00ef25581fc0303598c5e0acebf499e797c29e1e8810b2
contracts/utills/Adminable.sol	71dec912ec37f237dd1b91d3317859d825d93802905eda63fc04bc88d77155b5
contracts/voting/ApprovalVoting.sol	68dd752ee8d4d51e33b98ab2eb3caa548b56503d1b518f688c4cacbaa988d056
contracts/wallet_trading_limiter/helpers/TradingClassesMockup.sol	e825b520494ad3902a3f927a21aca0907ac392c6d3f12bc75ceb34eaf8bfe856
contracts/wallet_trading_limiter/helpers/WalletsTradingDataSourceExposure.sol	7524ebcabb223cb174f9df3814fca7b9ee815be84839751375d8cf357f1947e6
contracts/wallet_trading_limiter/helpers/WalletsTradingDataSourceMockup.sol	c0578e5cf7c46ccb285d61ca50868da7cea3854f6419a90c0c62d4d1f387b24f
contracts/wallet_trading_limiter/helpers/WalletsTradingLimiterMockup.sol	93e5d9a2a53e432e2289e0855d2f8b0d72d9653e2babd64c7880524c2d276671
contracts/wallet_trading_limiter/helpers/WalletsTradingLimiterValueConverterMockup.sol	590a19594abb7b1577795fdba6635e29265ead53bd566302c0bda6b42c64188
contracts/wallet_trading_limiter/interfaces/ITradingClasses.sol	de5169a71266121a6e026528a5160b14a9ed5b659718f812954392df100f7973
contracts/wallet_trading_limiter/interfaces/IWalletsTradingDataSource.sol	66963c85ee258c6337dceebb2ae041ce9bfa7c1e0711779a3c65f5e34463a6ad



contracts/wallet_trading_limiter/interfaces/IWalletsTradingLimiter.sol	052cdd7e298654a3c584114df5b8e5e92a31882b5f7c5f50e2d6c3c12dbf9e40
contracts/wallet_trading_limiter/interfaces/IWalletsTradingLimiterValueConverter.sol	6138f77f4961267be1d93420df90a6d14c9bc64e0beb9c31fe197dc1764a1578
contracts/wallet_trading_limiter/TradingClasses.sol	b06598c15c905acb79f04ca312824ca028cb5be0496a628b943ac65cd26e42d
contracts/wallet_trading_limiter/WalletsTradingDataSource.sol	d96d22bacd34f47b13edd8fc10667b144e9bb4ece0a267588d267969b8424f81
contracts/wallet_trading_limiter/WalletsTradingLimiterBase.sol	765c403f8a56c3600f7146a9c311335bef82ec81faca5a86948dceceacc32de
contracts/wallet_trading_limiter/WalletsTradingLimiterValueConverter.sol	4d437d475b7e59460140a2a68bbf1bac752f5c45831326d42c3cdad238b1accc
migrations/1_initial_migration.js	533b7003b8ea3b5249aa48706cb4078fc6c04ee4936436f17a550f36cd979ae3
migrations/2_deploy_contracts_phase1.js	f081d2e615ed4a76168dd9175782c7153ac3b5d4c634bce9f53dd5b0fecf8369
migrations/3_deploy_contracts_phase2.js	736c87d3e45e2ca2c679435798699bc71ba45453ec2e0a3999650caf88e300a0
migrations/4_initialize_model_data_source.js	a273f67b2b9f28442b55666f64a73532d64d1c09f9340657c1a1befd7ff2b608
migrations/util.js	92da65cbb5b7320709ab8cd022da5aa3429c7744d5135211efee3c6185d6e6e8
scripts/2_deploy_contracts_phase1.js	838163f1dc46d04efb6b75c6bc60561ef032b06bfff3dcf940e7859fe52500134
scripts/3_deploy_contracts_phase2.js	79006fee97f7733792d31dba09ad8fd2d052d4b56c0fece619560c9c1ce7840a
scripts/4_initialize_model_data_source.js	80fba8e0485a48c5051a7bf718af9b98787456b857afd51217398ff578916ac5
scripts/price-update/get_sdr_price.js	dcad41d6546f7ebc65d6ba92118792ba14084c16f100f5717b9609c56624ad3f
scripts/price-update/update_trading_converter.js	912afc5103b0c3a37261b0e9df9b0bd2fbc8a7938d463d8d44b21596dba3ae23
scripts/price-update/update_transaction_converter.js	7657d239977abfcba493b6cd95acce79d533006942100ad5d9353a9323eac124
scripts/util.js	42643558f57a3f8afb3eb8a60be31cb2457868872086997326d1e3eea071957f
test/authorization/AuthorizationDataSourceUnitTest.js	7522eb90f77dce40cecc95fed8baf25317039e472e5d00cf7d5b870827c69dda
test/authorization/helpers/AuthorizationActionRoles.js	39bc86339d4552b101c1adfb1ea79a213c011af7530d89e9f6bc1ad72c97bd13
test/contract_address_locator/ContractAddressLocatorHolderUnitTest.js	fcfe09fc0c7d909255ec22a3259124da4bea0235346d5c00f82068c19ebbfbf7
test/contract_address_locator/ContractAddressLocatorProxyUnitTest.js	1a0c4465588f8646c11859f4fb85c23b6b166e2bb4de0713c7b6753159e902f8
test/contract_address_locator/ContractAddressLocatorUnitTest.js	f27f68eaf9d0d9aa6f34f77722a544f8184c17dafbae496624e5d52fe206da42
test/exceptions.js	89eff46ad85c873ec1115790fb95933fe25ba52f5a6b2a53f73c1c9fc6eb866e
test/migrations/SGAToSGRInitializerUnitTest.js	33e39ffaa13bb30497559f27c965f99ff6e953aa270e3aef9b46f9cf9492e277
test/migrations/SGAToSGRTokenExchangeUnitTest.js	1c5e09bfcdece6ff139b4f0b30fa8a11ccf297365c8b8e26d6d457cc227b1bc5d
test/saga-genesis/helpers/SGNConversionManager.js	1f0507ed361d2a2d89e9e6f315f63a342e46e9e010975278e27ded0588636bec
test/saga-genesis/helpers/SGNToken.js	3b047422cb04ad4090769e1179d4852399e5e8535358e8ec79157deaeaccf9ec
test/saga-genesis/SGNAuthorizationManagerUnitTest.js	d8229c48e727745fa95b4e4dc9c0accf0ff306fb8815a2cb4823f0779fc1d40c
test/saga-genesis/SGNConversionManagerUnitTest.js	6949599402838cef211481d3c1ee8ec162a0e9d069528d1b2811c6d117f524c6
test/saga-genesis/SGNTokenFuncTest.js	9e9b141901c8570e35b63d9eb894f7526a00f9ac26db401efe8a16c21598320a
test/saga-genesis/SGNTokenManagerUnitTest.js	706f30d68869a285e70992045371908bbc2f57a26da7b6b01d1f062cb217ca0f
test/saga-genesis/SGNTokenUnitTest.js	0799f8ec97f7be19ad9c013c23471cc82672973420410673e076ef5651c392ec4
test/saga-genesis/SGNWalletsTradingLimiterUnitTest.js	0490d07327e7a3ee25038181462a3958c5345a631d432971f156a21f26d4b3a7
test/sogur/BatchSetModelDataSourceUnitTest.js	406e39232fb8b081b89bf2fdf82aeacc9c679ddfdaf35894f6b21c5164321508
test/sogur/ETHConverterUnitTest.js	759d9241b3f91d55254064d6ac2402d232772381a4f9d802216f6e1c8496f4f2
test/sogur/helpers/ModelDataSource.js	314c6cd751616b0b77232c657f775c6131ced0897e43944b8da7cd7564b9cdd9
test/sogur/helpers/sequences/BuyAll.js	c715fcbcee701d7dcbcc205c110a65669cfc8b255e4980ab94fb63b94dd767cb
test/sogur/helpers/sequences/BuyAllSellAll.js	4c528be047d78c8f471cd5579c215d12ce3fab21d76a90bfd7855075c38e6f82
test/sogur/helpers/sequences/BuySomeSellSome.js	998bba09e29bc6f90ea95aba895c03c59c4becaf429de8702eebd4653eb7a8f0
test/sogur/helpers/ShareHolderTestSequence.js	fae7cb3882be35f4f6bac64e111bcafc4df43b6b585b51540047260579c2d7d8
test/sogur/IntervalliteratorUnitTest.js	7949576ae61b0715f5e42ec51fadbf02052fda81241fab667b9ed62c990550df
test/sogur/MintingPointTimersManagerUnitTest.js	85e3a5957f7ee9f9713d676fb14f57aae84ac163cbcd4deebdb5c7f0de2ae430

test/sogur/MintManagerUnitTest.js	2572de57f993fb9361fe8168c3b19d9ff8be970506231583c39c315f98e2d43f
test/sogur/ModelCalculatorExtUnitTest.js	60c02c24858be51b88416ad9e98afc27808d559448b7a69cac280a9363eb2804
test/sogur/ModelCalculatorIntUnitTest.js	928e5035e80804643f59f56f58b0f74aeb5dd66094afa23a08d778cd2e5dfae0
test/sogur/ModelDataSourceUnitTest.js	5a514c7f7dadaeb62e3ddcc3dfd61f8f4a5176555fb33dc99473644ba143d9ed
test/sogur/MonetaryModelFuncTest.js	b6b3b88e647b90deb0120c2b5312a91c8a3f372e2f1b4ad3f7f56d81d63bf85d
test/sogur/MonetaryModelStateUnitTest.js	bbf47809f4e2179c645ea62061586965bf0e3d56be96bd0fdd65b4daf5ac5741
test/sogur/MonetaryModelUnitTest.js	eb5733873669ec70e25aac617acceaf1ed51be264c488454262a5421ccd655f
test/sogur/OracleRateApproverUnitTest.js	beb6506b95fd6ea2ead8a1f5b33989da0c657c28f5279a81667656e15788f992
test/sogur/PaymentManagerUnitTest.js	141845572a9bbef9f14ee62f0319961bcabd2b04520143614c6ba31b9b85f74f
test/sogur/PaymentQueueUnitTest.js	6f30c509c67d674e31dc34c49201ab24a12f694f4fe4019952cb6d61dc878938
test/sogur/PriceBandCalculatorExtUnitTest.js	517185b0e3d5070e314d9f09ff4bc934034930705e5ec0bd6c1166cdd2bef111
test/sogur/PriceBandCalculatorIntUnitTest.js	078daedad32474dfcaddc2c069c9e36066ba97271e674f951ddd32b192400215
test/sogur/ReconciliationAdjusterUnitTest.js	86a31978bdefa56063591ade4707f178d3a808721f36b7ece116d8bf5121c6e6
test/sogur/RedButtonUnitTest.js	c93b89b51a5c865622fc777afec39c48ea165f12f0f04edade385df6449dd14b
test/sogur/ReserveManagerUnitTest.js	6078d38d603d5964fedf25e0983632b928ebb15ba324a9b41acd108f124b955f
test/sogur/SGRAuthorizationManagerUnitTest.js	1838a951e93467bb8e8c8d42858cc942a6ab52253bbe45896a7e81a43793cdbc
test/sogur/SGRTTokenFuncTest.js	bb0e88fea6be03569b48b31f52ea75bd02eaa2c383b51e0c584615b31ec01621
test/sogur/SGRTTokenManagerUnitTest.js	5479866c3adaeb4a40d71b345d029fb2d8d33a6411ee6a35a57bb0c9dd27c07e
test/sogur/SGRTTokenUnitTest.js	f4536971edf1119cbfc95d3c2ac19891bfef4ca6a5b6bec1154aaf94f865700
test/sogur/SGRWalletsTradingLimiterUnitTest.js	39bc8081ea1b7e5a8cfaa2bc5f19c83c82135bed100876a14d5f0c039cb23252
test/sogur/ShareHolderTest.js	ec6374b051a2dad9d340b40cf7d651cec31763a48312ea09c315f372e7c610b6
test/sogur/SystemTest.js	75ab8b4d414e54452f22c38eb92eb9656d9bdd1e4c44b0d7957a935d4d0aef44
test/sogur/TransactionLimiterUnitTest.js	9d03dd8c0446481667eeddf9107cda002dab1164991831d0aca3b0b735ee0a3d
test/sogur/TransactionManagerUnitTest.js	2abbe1759b3849551c5ec56825353bddace15243460b632ea8ead4e8320eb7ab
test/utilities.js	8e0d4ef0621645f0f2029367870dbb61564008c01b091b84a253fa05b0172134
test/utlis/AdminableUnitTest.js	14ae f8c8920bb402c59eae15d22359cd006be20b004f4169184207bcd84dccc
test/voting/ApprovalVotingUnitTest.js	9309dc2aa509425b4505613d3d9846511530fc76a0c144b5de638738e5939867
test/wallet_trading_limiter/TradingClassesUnitTest.js	893b812d7330e1aa01b2749399d2e0fc325015c98997c66acd66e40a41645a61
test/wallet_trading_limiter/WalletsTradingDataSourceUnitTest.js	efe79889b5b9420107c938cfd11ce5df0b9159fa8ccac1c7c7fb7f01fc4c871e
test/wallet_trading_limiter/WalletsTradingLimiterValueConverterUnitTest.js	fd502db3db83296687881bb56730417bde0202f7837cd22cd4261e580c2f8a2
truffle-config.js	dad238dcd96490913105ff5cca9201aa68d2d5c2059aa7553f93169f59cc3889
.solcover.js	47c998a2eef89c08fb4f7de0b9604ff4c40953f1ce43142e5b24ba1cc02cf55c

## Appendix B: Natural Logarithm Tests

Described below is the precision testing of the natural logarithm approximation using Taylor series with fixed-point operations, as it was implemented in `contracts/sogur/PriceCalculator.sol`:

Input	Control	PriceCalculator.sol	Change (%)
1	0.0000000000000000	0.0000000000000000	0.000000000000000%
1.19	0.173953307123438	0.173953307123438	0.000000000000000%
1.38	0.322083499169113	0.322083499169113	-0.000000000000001%
1.57	0.451075619360217	0.451075619360216	-0.000000000000002%
1.76	0.565313809050061	0.565313809050060	-0.000000000000001%
1.95	0.667829372575655	0.667829372575655	-0.000000000000001%
2.14	0.760805829033760	0.760805829033760	0.000000000000000%
2.33	0.845868267577609	0.845868267577609	0.000000000000000%
2.52	0.924258901523332	0.924258901523331	-0.000000000000001%
2.71	0.996948634891610	0.996948634891609	0.000000000000000%
2.9	1.064710736992430	1.064710736992420	-0.000000000000008%
3.09	1.128171090909650	1.128171090909650	-0.000000000000004%
3.28	1.187843422396050	1.187843422396050	-0.000000000000002%
3.47	1.244154593958770	1.244154593958760	-0.000000000000006%
3.66	1.297463147413280	1.297463147413270	-0.000000000000004%
3.85	1.348073148299690	1.348073148299690	-0.000000000000002%

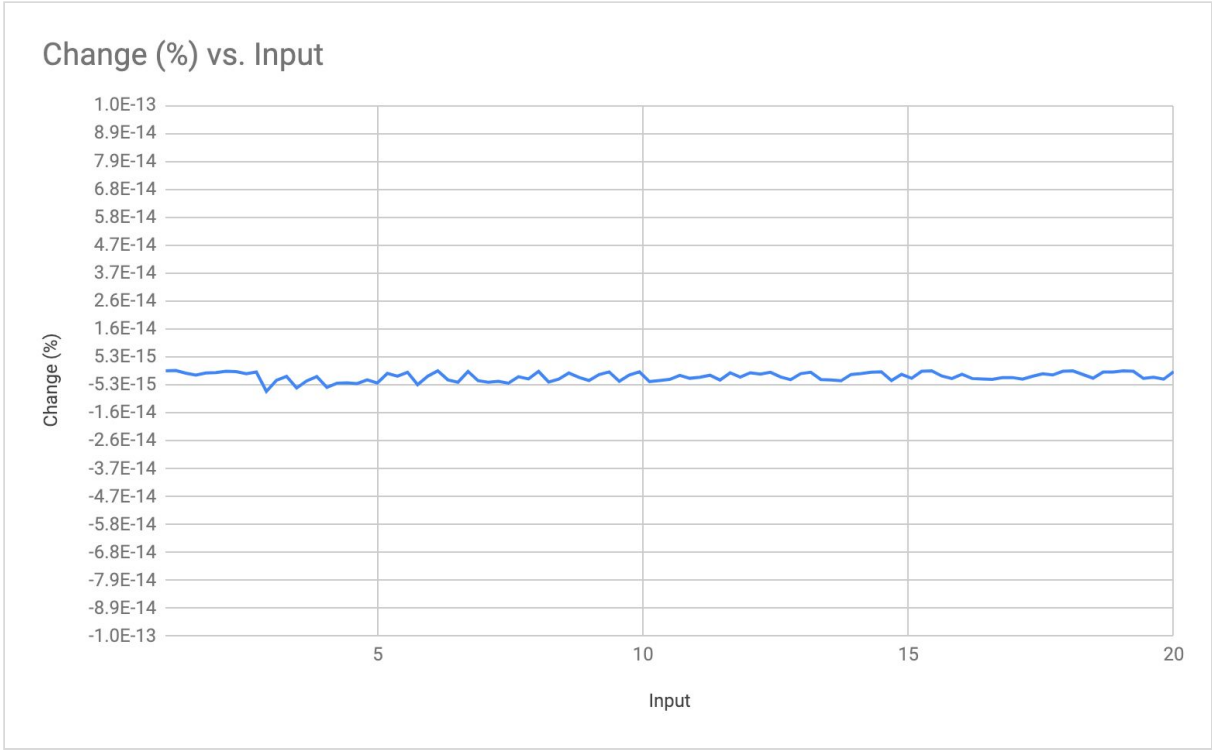
4.04	1.396244691973060	1.396244691973050	-0.00000000000006%
4.23	1.442201993058190	1.442201993058180	-0.00000000000005%
4.42	1.486139696089610	1.486139696089600	-0.00000000000004%
4.61	1.528227857008560	1.528227857008550	-0.00000000000005%
4.8	1.568615917913850	1.568615917913840	-0.00000000000003%
4.99	1.607435909763430	1.607435909763420	-0.00000000000005%
5.18	1.644805056271390	1.644805056271390	-0.00000000000001%
5.37	1.680827908520770	1.680827908520770	-0.00000000000002%
5.56	1.715598108262490	1.715598108262490	-0.00000000000001%
5.75	1.749199854809260	1.749199854809250	-0.00000000000005%
5.94	1.781709133374550	1.781709133374550	-0.00000000000002%
6.13	1.813194749948120	1.813194749948120	0.00000000000000%
6.32	1.843719208158770	1.843719208158760	-0.00000000000003%
6.51	1.873339456220480	1.873339456220470	-0.00000000000004%
6.7	1.902107526396920	1.902107526396920	0.00000000000000%
6.89	1.930071085025570	1.930071085025560	-0.00000000000004%
7.08	1.957273907705630	1.957273907705620	-0.00000000000004%
7.27	1.983756291545430	1.983756291545420	-0.00000000000004%
7.46	2.009555414215670	2.009555414215660	-0.00000000000005%
7.65	2.034705647838440	2.034705647838440	-0.00000000000002%
7.84	2.059238834362320	2.059238834362310	-0.00000000000003%

8.03	2.083184527958670	2.083184527958670	0.00000000000000%
8.22	2.106570209068090	2.106570209068080	-0.00000000000004%
8.41	2.129421473984860	2.129421473984850	-0.00000000000003%
8.6	2.151762203259460	2.151762203259460	-0.00000000000001%
8.79	2.173614711697090	2.173614711697080	-0.00000000000002%
8.98	2.194999882314110	2.194999882314100	-0.00000000000004%
9.17	2.215937286268370	2.215937286268370	-0.00000000000001%
9.36	2.236445290489500	2.236445290489500	0.00000000000000%
9.55	2.256541154492640	2.256541154492630	-0.00000000000004%
9.74	2.276241117654440	2.276241117654440	-0.00000000000002%
9.93	2.295560478057080	2.295560478057080	0.00000000000000%
10.12	2.314513663859320	2.314513663859310	-0.00000000000004%
10.31	2.333114298028870	2.333114298028860	-0.00000000000004%
10.5	2.351375257163480	2.351375257163470	-0.00000000000003%
10.69	2.369308725036950	2.369308725036950	-0.00000000000002%
10.88	2.386926241427800	2.386926241427790	-0.00000000000003%
11.07	2.404238746720550	2.404238746720540	-0.00000000000002%
11.26	2.421256622711540	2.421256622711540	-0.00000000000002%
11.45	2.437989730000250	2.437989730000240	-0.00000000000003%
11.64	2.454447442303290	2.454447442303290	-0.00000000000001%
11.83	2.470638677990300	2.470638677990290	-0.00000000000002%

12.02	2.486571929107060	2.486571929107060	-0.00000000000001%
12.21	2.502255288122610	2.502255288122610	-0.00000000000001%
12.4	2.517696472610990	2.517696472610990	-0.00000000000001%
12.59	2.532902848056260	2.532902848056250	-0.00000000000002%
12.78	2.547881448949390	2.547881448949380	-0.00000000000003%
12.97	2.562638998328350	2.562638998328350	-0.00000000000001%
13.16	2.577181925897170	2.577181925897170	-0.00000000000001%
13.35	2.591516384846260	2.591516384846250	-0.00000000000003%
13.54	2.605648267484130	2.605648267484120	-0.00000000000003%
13.73	2.619583219779880	2.619583219779870	-0.00000000000004%
13.92	2.633326654906270	2.633326654906270	-0.00000000000001%
14.11	2.646883765864720	2.646883765864720	-0.00000000000001%
14.3	2.660259537265860	2.660259537265860	-0.00000000000001%
14.49	2.673458756332590	2.673458756332590	0.00000000000000%
14.68	2.686486023186370	2.686486023186360	-0.00000000000004%
14.87	2.699345760472060	2.699345760472060	-0.00000000000001%
15.06	2.712042222371750	2.712042222371740	-0.00000000000003%
15.25	2.724579503053420	2.724579503053420	0.00000000000000%
15.44	2.736961544596630	2.736961544596630	0.00000000000000%
15.63	2.749192144433390	2.749192144433380	-0.00000000000002%
15.82	2.761274962339510	2.761274962339500	-0.00000000000003%

16.01	2.773213527008620	2.773213527008620	-0.00000000000001%
16.2	2.785011242238340	2.785011242238330	-0.00000000000003%
16.39	2.796671392755740	2.796671392755730	-0.00000000000003%
16.58	2.808197149707150	2.808197149707140	-0.00000000000003%
16.77	2.819591575835120	2.819591575835110	-0.00000000000003%
16.96	2.830857630363760	2.830857630363750	-0.00000000000003%
17.15	2.841998173611950	2.841998173611940	-0.00000000000003%
17.34	2.853015971352400	2.853015971352390	-0.00000000000002%
17.53	2.863913698933140	2.863913698933140	-0.00000000000001%
17.72	2.874693945176930	2.874693945176930	-0.00000000000002%
17.91	2.885359216072620	2.885359216072620	0.00000000000000%
18.1	2.895911938271780	2.895911938271780	0.00000000000000%
18.29	2.906354462402770	2.906354462402770	-0.00000000000001%
18.48	2.916689066213540	2.916689066213530	-0.00000000000003%
18.67	2.926917957553630	2.926917957553630	0.00000000000000%
18.86	2.937043277205310	2.937043277205310	0.00000000000000%
19.05	2.947067101572710	2.947067101572710	0.00000000000000%
19.24	2.956991445237560	2.956991445237560	0.00000000000000%
19.43	2.966818263389350	2.966818263389340	-0.00000000000003%
19.62	2.976549454137220	2.976549454137210	-0.00000000000002%
19.81	2.986186860710460	2.986186860710450	-0.00000000000003%

20	2.995732273553990	2.995732273553990	0.00000000000000%
----	-------------------	-------------------	-------------------



Needless to say that the estimation method in **contracts/PriceCalcualtor.sol** is excellent.



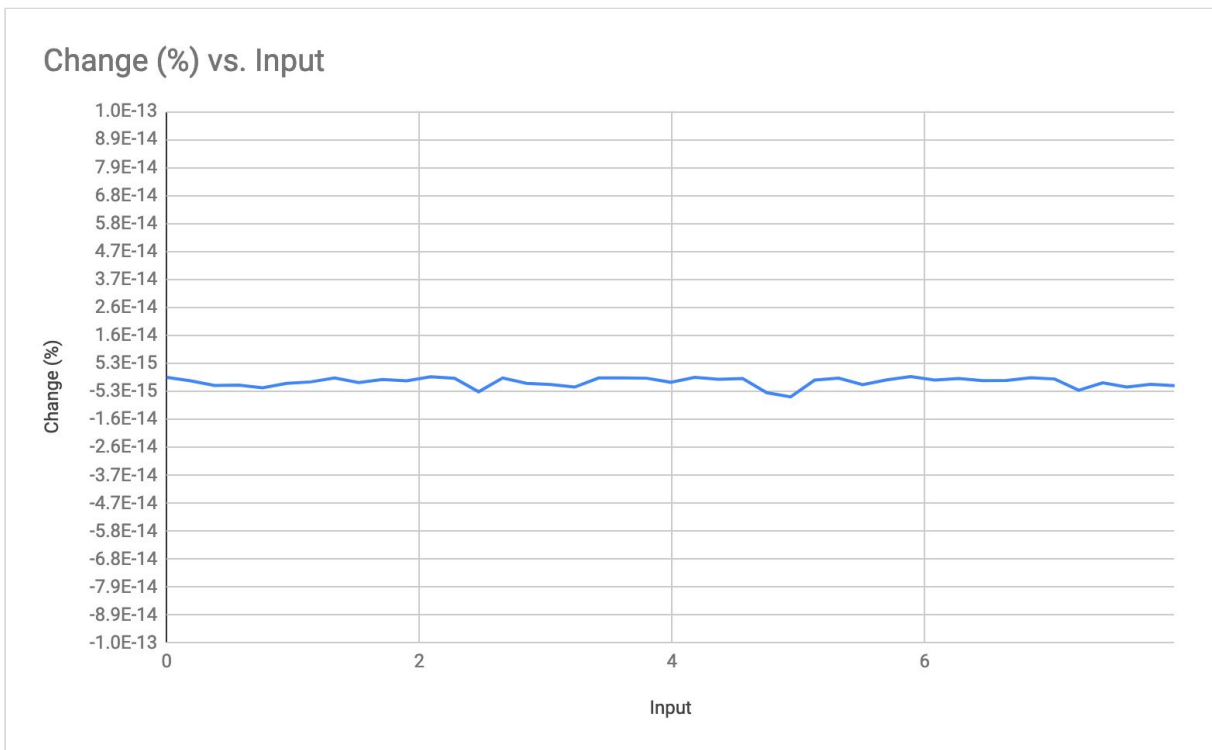
## Appendix C: Exponential Function Tests

Described below is the precision testing of the exponential function approximation using Taylor series with fixed-point operations, as it was implemented in **contracts/sogur/PriceCalculator.sol**:

Input	Control	PriceCalculator.sol	Change (%)
0	1.0000000000000000	1.0000000000000000	0.0000000000000000
0.19	1.209249597657250	1.209249597657250	-0.0000000000000001
0.38	1.462284589434220	1.462284589434220	-0.0000000000000003
0.57	1.768267051433740	1.768267051433730	-0.0000000000000003
0.76	2.138276220496820	2.138276220496810	-0.0000000000000004
0.95	2.585709659315850	2.585709659315840	-0.0000000000000002
1.14	3.126768365186160	3.126768365186150	-0.0000000000000002
1.33	3.781043387568780	3.781043387568780	0.0000000000000000
1.52	4.572225195142160	4.572225195142150	-0.0000000000000002
1.71	5.528961477624000	5.528961477624000	-0.0000000000000001
1.9	6.685894442279270	6.685894442279260	-0.0000000000000001
2.09	8.084915164305060	8.084915164305060	0.0000000000000000
2.28	9.776680409528900	9.776680409528900	0.0000000000000000
2.47	11.822446851646400	11.822446851646300	-0.0000000000000005
2.66	14.296289098677600	14.296289098677600	0.0000000000000000
2.85	17.287781840567600	17.287781840567600	-0.0000000000000002

3.04	20.905243235092800	20.905243235092700	-0.0000000000000003
3.23	25.279656970962900	25.279656970962800	-0.0000000000000004
3.42	30.569415021050200	30.569415021050200	0.0000000000000000
3.61	36.966052814822500	36.966052814822500	0.0000000000000000
3.8	44.701184493300800	44.701184493300800	0.0000000000000000
3.99	54.054889363326600	54.054889363326500	-0.0000000000000002
4.18	65.365853214009900	65.365853214009900	0.0000000000000000
4.37	79.043631699564500	79.043631699564400	-0.0000000000000001
4.56	95.583479830066200	95.583479830066200	0.0000000000000000
4.75	115.584284527188000	115.584284527187000	-0.0000000000000006
4.94	139.770249560003000	139.770249560002000	-0.0000000000000007
5.13	169.017118044887000	169.017118044887000	-0.0000000000000001
5.32	204.383881992968000	204.383881992968000	0.0000000000000000
5.51	247.151127067624000	247.151127067623000	-0.0000000000000003
5.7	298.867400967060000	298.867400967060000	-0.0000000000000001
5.89	361.405284372286000	361.405284372286000	0.0000000000000000
6.08	437.029194718391000	437.029194718391000	-0.0000000000000001
6.27	528.477377877687000	528.477377877687000	0.0000000000000000
6.46	639.061056569553000	639.061056569552000	-0.0000000000000001
6.65	772.784325535150000	772.784325535149000	-0.0000000000000001
6.84	934.489134729210000	934.489134729210000	0.0000000000000000

7.03	1130.030610186370000	1130.030610186370000	-0.0000000000000001
7.22	1366.489060708250000	1366.489060708240000	-0.0000000000000005
7.41	1652.426346864480000	1652.426346864480000	-0.0000000000000002
7.6	1998.195895104120000	1998.195895104110000	-0.0000000000000004
7.79	2416.317582195030000	2416.317582195020000	-0.0000000000000003
7.98	2921.931064081480000	2921.931064081470000	-0.0000000000000003



## Appendix D: SECURITY.md Template<sup>5</sup>

### Security Policy

[COMPANY] is committed to resolving security vulnerabilities in our software quickly and carefully. We take the necessary steps to minimize risk, provide timely information, and deliver vulnerability fixes and mitigations required to address security issues.

### Reporting a Vulnerability

Security vulnerabilities in [PROJECT] software should be reported by email to [SECURITY\_EMAIL]. If you think your report might be eligible for the [PROJECT] Bug Bounty Program, your email should be sent to [BOUNTY\_EMAIL].

Your report should include the following:

- Your name (optional).
- Description of the vulnerability.
- Attack scenario or a reproduction.
- Any other details.

Try to include as much information in your report as you can, including a description of the vulnerability, its potential impact, and steps for reproducing it. Be sure to use a descriptive subject line.

You'll receive a response to your email within [X] business days, indicating the next steps in handling your report. We encourage finders to use encrypted communication channels to protect the confidentiality of vulnerability reports. You can encrypt your report using our public key. This key is [PUBLIC\_PGP\_REGISTRY] server and reproduced below.

After the initial reply to your report, our team will endeavor to keep you informed of the progress being made towards a fix. These updates will be sent at least every [X] business days.

Thank you for taking the time to disclose any vulnerabilities you find responsibly.

### Responsible Investigation and Reporting

Responsible investigation and reporting include, but isn't limited to, the following:

- Don't violate the privacy of other users, destroy data, etc.

---

<sup>5</sup> Adapted from <https://github.com/paritytech/parity-ethereum/blob/master/SECURITY.md>

- Don't defraud or harm Parity Technologies Ltd or its users during your research; you should make a good faith effort not to interrupt or degrade our services.
- Don't target our physical security measures or attempt to use social engineering, spam, distributed denial of service (DDOS) attacks, etc.
- Initially, report the bug only to us and not to anyone else.
- Give us a reasonable amount of time to fix the bug before disclosing it to anyone else, and give us adequate written warning before disclosing it to anyone else.
- In general, please investigate and report bugs in a way that makes a reasonable, good-faith effort not to be disruptive or harmful to our users or us. Otherwise, your actions might be interpreted as an attack rather than an effort to be helpful.

## Bug Bounty Program

Our Bug Bounty Program allows us to recognize and reward members of the [PROJECT] community for helping us find and address significant bugs, in accordance with the terms of the [PROJECT] Bug Bounty Program. A detailed description of eligibility, rewards, legal information and terms & conditions for contributors can be found on [WEBSITE].

## Plaintext PGP Key

-----BEGIN PGP PUBLIC KEY BLOCK-----

```
mQMubExtIfMRCACe8xHPWBciPfCkdN+4TNUPW04ahDd1SJk5fmzaFcx8GnJILvzt
+0Vbs4HPLUj0yJQz0ZXz+8EPqzs/sqBYZjh5doyGFqXG/Q3oD4Yxru9+msvTQsd4
yWQU1+2C/wYcomhHp3pRRbbLPn4/UYx0Qt0eo0P5inr9DIPzQ4Ejz744DSSR5SAN
AuYhFqN6Xx0qQ04Rxd1vxMQG3KIVsFGZ5IUBaY3Zp10/u165nYgyE9jujaBBF106
0vEIUIHE0CwHReCsNSDZgS891WfMDKFBcmfCP7hGfHE2s0WcTGVC38aaPjFQe76c
WIXp191xySc71qVUeuKMWBLWLwSX+jZCFxWDAQDzDYJSwjdDvtGvF1X2ddcsylki
/pK7uz5E4AuVzxoasQf/WfSpc6VfWPIXkbPvstGq8sxx4cClY1ur11bpQ8ZsqzSL
EBOK2v+f2wnRGJAXy+Jq3Ur+mQhfZJiq+sM4gCCcV19/uKnQrGWYKnZ8E9v3q0ot
09i0/23HX9oFK8A11q6eJpXF89Y1cUwHVmAGaDoqMEc00vHYSZc+TMfzDR1PW9AD
08wzkL1FYzSX8/5iV73PPpy1K4QCecKt6ejXg85WpEb14HCBXotcQLL9J0+NFbaX
```

...

...

...

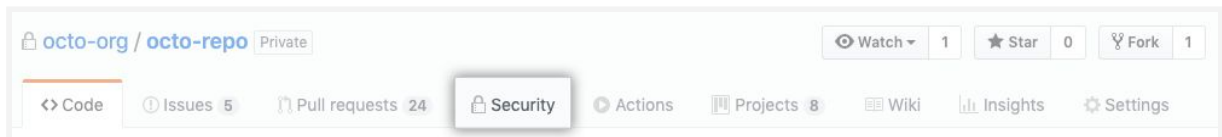
```
SpP2Mu79K6Rf2zZFUV4frPhuKxTtftXazLj0YbzEbIWn5l2z2knjYX8rFRZyAAD+
MVRhQqRvqK2lMH+JSYoir1A5rmiFOWeiCFayTsBE0WwA/R3ZgoH2n/lxtxyBDUmb
18Bd8Kck8P5qEEeYI+M0NbpH
=iyJr
```

-----END PGP PUBLIC KEY BLOCK-----

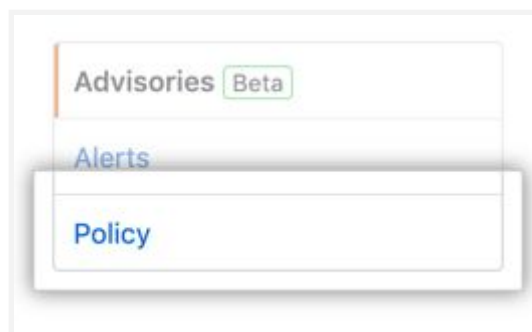
## Appendix E: Setting up Github Security Policy

To set up Github Security Policy, you should:

- On GitHub, navigate to the main page of the repository.
- Under your repository name, click **Security**:



- In the left sidebar, click **Policy**:



- Click **Start Setup**:

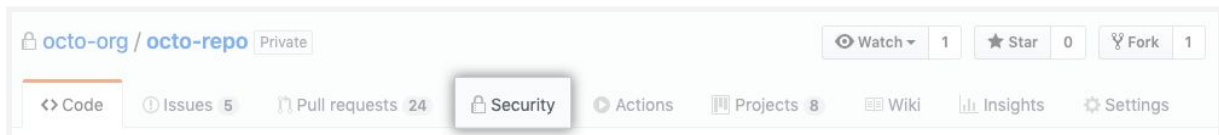


- Add and commit your SECURITY.md.

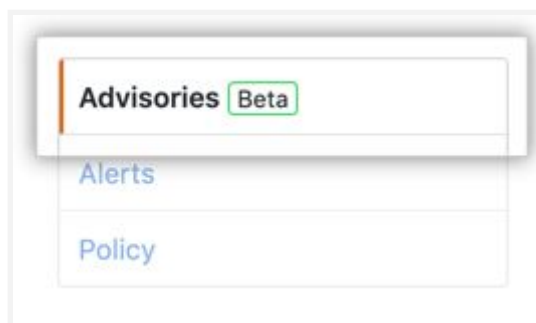
## Appendix F: Setting up Maintainer Security Advisory

Anyone with admin permissions to a repository can create a security advisory:

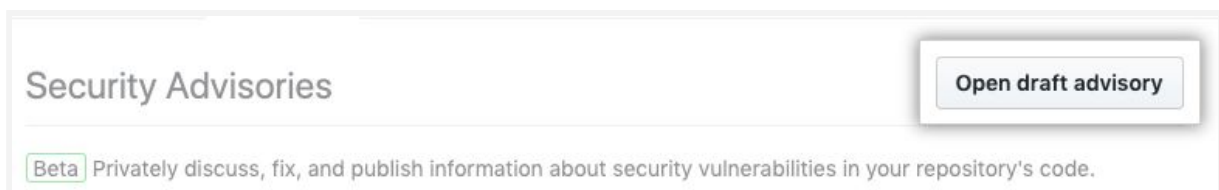
- On GitHub, navigate to the main page of the repository.
- Under your repository name, click **Security**:



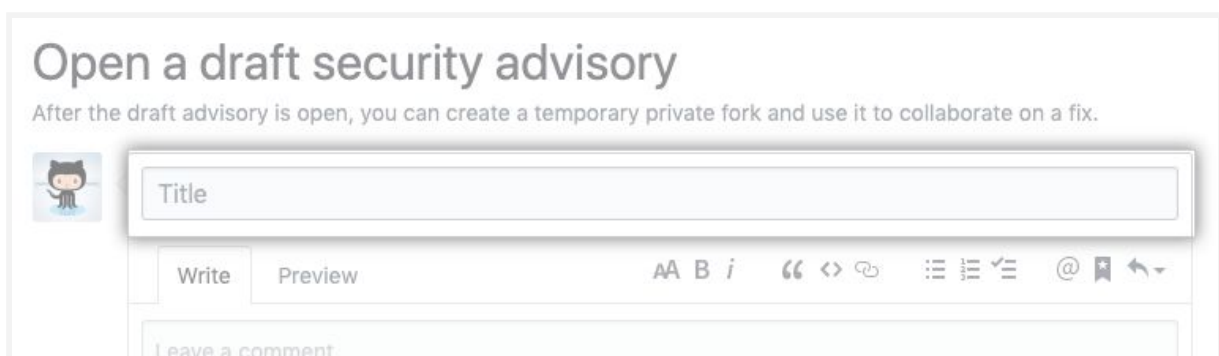
- In the left sidebar, click **Advisories**:



- Click **Open Draft Advisory**:



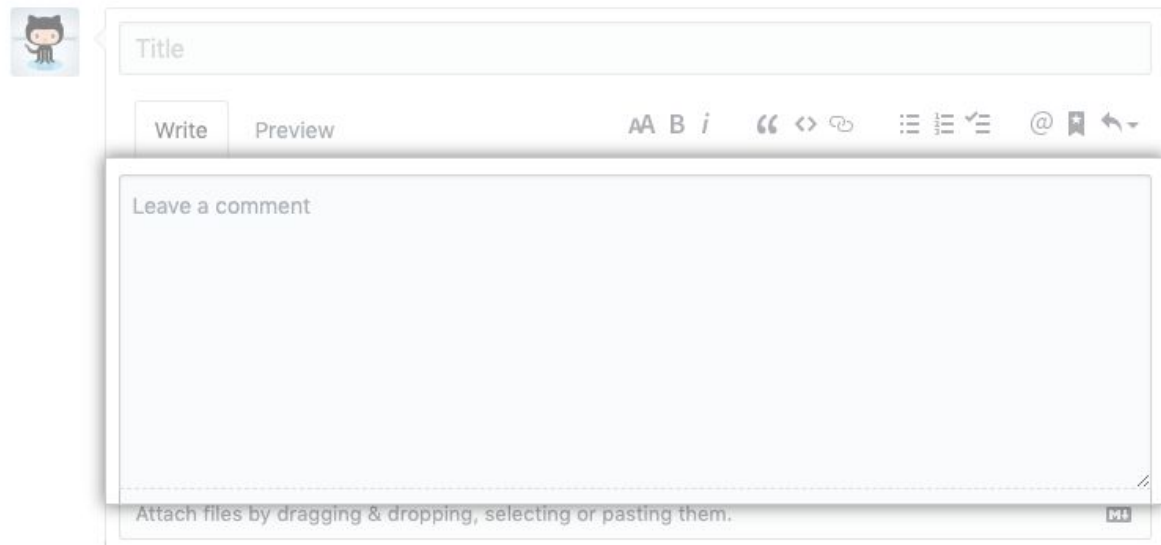
- Type a title for your security advisory:



- Type a description of the security vulnerability.

## Open a draft security advisory

After the draft advisory is open, you can create a temporary private fork and use it to collaborate on a fix.



The screenshot shows the GitHub interface for creating a draft security advisory. On the left is the GitHub logo. To its right is a text input field labeled "Title". Below the title field are two tabs: "Write" (which is active) and "Preview". To the right of the tabs is a rich text editor toolbar with icons for bold (AA), italic (i), quote, code (<>), link, list, ordered list, checkmark, mention (@), bookmark, and undo. Below the toolbar is a large text area with the placeholder text "Leave a comment". At the bottom of the text area is a dashed line and the text "Attach files by dragging & dropping, selecting or pasting them." followed by a small "M4" icon.

- Click **Create Draft Advisory**.