


Uno Re V2

Smart Contract Audit Report

AUDIT SUMMARY

 **UnoRe** is building a set of smart contracts to facilitate a single-sided insurance and reinsurance platform with opportunities to stake funds and earn rewards. This is the updated second iteration of the platform.

We previously reviewed the project team's V1 [here](#).

AUDIT SCOPE

Name	Source Code	Visualized
SingleSidedInsurancePool	2ca5fb5	Inheritance Chart. Function Graph.
SingleSidedReinsuracePool	2ca5fb5	Inheritance Chart. Function Graph.
RiskPoolFactory	2ca5fb5	Inheritance Chart. Function Graph.
RiskPool	2ca5fb5	Inheritance Chart. Function Graph.
RewarderFactory	2ca5fb5	Inheritance Chart. Function Graph.
Rewarder	2ca5fb5	Inheritance Chart. Function Graph.
CapitalAgent	2ca5fb5	Inheritance Chart. Function Graph.
SalesPolicyFactory	2ca5fb5	Inheritance Chart. Function Graph.
SalesPolicy	2ca5fb5	Inheritance Chart. Function Graph.
PremiumPool	2ca5fb5	Inheritance Chart. Function Graph.
ExchangeAgent	2ca5fb5	Inheritance Chart. Function Graph.
EscalationManager	2ca5fb5	Inheritance Chart. Function Graph.
ClaimProcessor	2ca5fb5	Inheritance Chart. Function Graph.
MultiSigWallet	2ca5fb5	Inheritance Chart. Function Graph.

AUDIT FINDINGS

No findings were identified, though some centralized aspects are present.

Date: December 29th, 2023.

CONTRACTS OVERVIEW

- The contracts utilize `ReentrancyGuard` to protect against reentrancy attacks in applicable functions.
- As the contracts are implemented with Solidity v0.8, they are safe from any possible overflows/underflows.
- The team must exercise caution when assigning the staking tokens to avoid using fee-on-transfer tokens.

SingleSidedInsurancePool & SingleSidedReinsuracePool:

- These contracts allow users to stake a single asset determined by the project team into a Risk Pool configured by the project team in exchange for rewards from a Rewarder contract also configured by the project team.
- The SSIP supports ETH as an asset but the SSRP does not.
- The funds deposited into the SSIP are tracked by the Capital Agent in order to meet the platform's minimum liquidity requirements in preparation for future policy claims. The SSRP does not use the Capital Agent.
- Users may stake the Risk Pool's currency when the staking start time has passed and the contract is not "killed".
- The staked currency will be transferred to the Risk Pool address. The user will be minted Risk Pool tokens based on the staked amount and current "lp Price".
- In order for a user to exit their staked position, users must submit a withdrawal request and wait a lock period in order to withdraw their funds; rewards continue to accrue during the waiting period.
- By default, the lock period is 10 days.
- Once the wait time has elapsed, users can also call the `migrate()` function to trigger a migration of their underlying assets in the Risk Pool to the "migrateTo" address set by the project team.
- The `migrate()` function subsequently calls the `onMigration()` function on the Migration contract. This contract was not provided to us within the scope of this audit, so we are unable to provide an assessment in terms of security.
- Users may cancel their withdrawal requests at any time.
- Users earn rewards per block based on the amount of currency they have staked relative to the total amount staked and the "multiplier per block" value.
- Rewards are automatically calculated and transferred on withdrawals, but any user may also choose to harvest their rewards manually once the staking start time has passed.
- Any address with the Bot role may trigger a reward "roll over" if the Risk Pool and Reward contracts have the same currency.
- This will stake and transfer users' pending rewards to the Risk Pool if they have not enabled "not roll over".
- Users may toggle their not roll over at any time.
- In the SSIP contract the owner of a non-expired Policy, as determined in the Capital Agent contract, may request a payout up to the Policy's coverage amount if the contract's "failed" value has not been enabled.
- This function relies on an Oracle to execute; the Oracle was not provided to us in the scope of the audit, so we are unable to provide an assessment in terms of security.
- A request will be sent to the Claim Processor contract if the fail value has been enabled.
- The Claim Processor contract may "settle" a payout after it has been requested. This will trigger a "policy claim" in the Risk Pool and Capital Agent contracts.
- Both contracts also manage users' reward debt and pending rewards when Risk Pool shares are transferred. Shares pending withdrawal may not be transferred
- The sender will have their rewards claimed and both users' reward debt are updated.
- In the SSRP contract any address with the Claim Accessor role can call the `policyClaim()` function to trigger a policy claim in the contract's Risk Pool.
- Any address with the Guardian Council role may grant another address a role. Addresses granted a role through this method may not perform a privileged action until their "role lock time" has passed.
- Any address with the Admin role may use this contract to create a new Risk Pool via a specified Risk Pool Factory.
- Any address with the Admin role may use this contract to create a Rewarder via a specified Rewarder Factory.
- Any address with the Admin role may use this contract to create a SyntheticSSIP/SyntheticSSRP via a specified SyntheticSSIP Factory or SyntheticSSRP Factory.
- Any address with the Admin role may pause and unpause the contract.
- Any address with the Admin role may "kill" and "revive" the SSIP contract.
- Any address with the Guardian Council role may toggle the failed status in the SSRP contract.
- Any address with the Admin role may set the Escalation Manager, Claim Processor, Capital Agent addresses in the SSIP contract.

- Any address with the Admin role may set the migrateTo address.
- Any address with the Admin role may set the reward multiplier to any non-zero value.
- Any address with the Admin role may use this contract to set the min LP capital to any non-zero value in the Risk Pool contract.
- Any address with the Admin role may set the lock time to any non-zero value.
- Any address with the Admin role may set the assertion live time to any non-zero value.
- Any address with the Admin role may set the staking start time to any future timestamp.

RiskPool & RiskPoolFactory Contract:

- Anyone can use the RiskPoolFactory contract to create new RiskPool instances at any time.
- The RiskPool contract is used to store a specified currency in return for Risk Pool shares. The team should ensure the currency has 18 decimals to ensure proper conversion between staked assets and Risk Pool shares.
- All non-ERC20 functions in this contract may only be called by the SSRP address.
- Upon entering the Risk Pool, users are minted an amount of shares based on the "LP price". Initially, the price is 1 asset token for 1 share, but this is subject to change as tokens are removed from the Pool via a "policy claim".
- After a policy claim, the share price is recalculated based on the proportion of tokens in the contract to shares in circulation.
- Users may transfer their available Risk Pool tokens to any address, and the underlying asset is reassigned to the recipient through the SSRP contract.
- In order for a user to exit their staked position in the Risk Pool, users must submit a withdrawal request using the SSRP contract and wait a 10 day lock period in order to withdraw their funds.
- Once the wait time has elapsed, users can exchange their shares for an amount of the asset token based on the current share price.
- Users may cancel their withdrawal requests at any time.
- Alternatively, users can use the migrate() function in the SingleSidedReinsurancePool contract to withdraw their staked assets.
- Users must exercise caution when using the migrate() function as all shares the user are holding will be burned and their underlying value will be sent to the Migration contract controlled by the project team.
- The SSRP address may perform a policy claim at any time.
- This will transfer the specified amount of the underlying asset to the specified address.
- If the transfer would cause the contract's balance to fall below the "min LP" value, the contract will retain the min LP balance and transfer the remaining currency.
- The SSRP address may set the min LP to any non-zero value at any time.

Rewarder & RewarderFactory Contract:

- Anyone can use the RewarderFactory contract to create new Rewarder instances at any time.
- The Rewarder contract is used to facilitate staking rewards within the SingleSidedInsurancePool and SingleSidedReinsurancePool contracts.
- Only a single reward asset is supported by this contract.
- Upon reward distribution, the SSIP or SSRP pool assigned to this contract will trigger the onReward() function, which will attempt to transfer the requested amount of the reward token to the specified address.
- The specified address must have initiated the transaction and the contract must not be paused in order for rewards to be distributed.
- This contract must have a sufficient amount of rewards tokens supplied to it for reward distribution to be successful.
- The Operator address may withdraw any reward tokens from the contract at any time; note that any other token or ETH is not able to be withdrawn.
- The Operator address may transfer the Operator Role to any address at any time.
- The Operator address may pause and unpause the contract at any time.

CapitalAgent Contract:

- This contract is used to manage the total capital staked within SingleSidedInsurancePool contracts and the total amount utilized by all the coverage policies in the SalesPolicy contract.
- The contract uses the Exchange Agent to convert the capital of each SSIP to its value in USDC.
- Each policy calls this contract during deposits and withdraws to appropriately maintain the amount of capital in each policy.
- The contract maintains a minimum capital ratio (MCR), which is checked whenever a user attempts to unstake from an SSIP; the user can never withdraw more than a certain percentage of the total capital staked across all SSIPs as determined by the team.
- Each SSIP has an individual solvency capital requirement (SCR) checked whenever a user attempts to unstake from an SSIP; the user can never withdraw more than a certain percentage of the capital staked within the SSIP being withdrawn from as determined by the team.
- The contract also maintains a maximum leverage ratio (MLR), which dictates what percentage of the total capital staked can be utilized for policy coverage.

- The MLR is checked whenever a user attempts to purchase coverage for a policy; the ratio of total utilized amount to total capital staked can never be greater than the MLR set by the team.
- These restrictions do not apply to policy claims, although the total utilized amount values are appropriately deducted.
- Any address may update a policy's status. If the policy's duration has elapsed it will be marked as expired and the utilized amount removed from the total.
- The owner may set the Sales policy Factory and Operator addresses at any time.
- The owner may add and remove a SSIP from the whitelist at any time.
- The owner may add and remove a SSIP from the contract at any time.
- An address on the SSIP whitelist may add a SSIP to the contract at any time.
- The Sales Policy Factory and the owner may create the information tracked by this contract for a specified policy at any time.
- The owner may remove the information tracked by this contract for a specified policy at any time.
- The operator address may set the MCR and MLR percentages to any non-zero value at any time.
- The operator address may set the SCR percentage for a specified SSIP at any non-zero value at any time.
- The owner may set the Exchange Agent address used to determine token value in USDC at any time.

SalesPolicyFactory:

- The owner can use this contract to deploy and manage a single SalesPolicy instance and various protocol addresses.
- The owner may set the PremiumPool and ExchangeAgent contract addresses within the SalesPolicy contract at any time.
- The owner may set the signer address and the deadline for buying a policy to any value at any time.
- The owner may set the protocol URI value in the SalesPolicy to any value at any time.
- The owner may add any token address as an approved premium currency in the Sales Policy contract at any time.
- The owner may also use this contract to manage a list of Protocols; each protocol has an address and a blacklist status.
- The owner may add a new protocol address at any time.
- The owner can blacklist a Protocol at any time; blacklisted Protocols cannot be removed from the blacklist, but the owner can create a new Protocol with the same address, overwriting the previously stored data.
- The owner may use this contract to control whether or not the blacklist is checked within the SalesPolicy contracts when users try to buy a policy for a specific Protocol.
- The owner may set the CapitalAgent contract address used by the SalesPolicy to any address at any time.
- The owner may use this contract to give a currency address a maximum approval in a Protocol at any time.

SalesPolicy Contract:

- Anyone can use this contract to buy purchase coverage offered by the platform when the contract is not paused.
- Users are able to specify their own coverage amount, duration, and policy price for one or many Protocols.
- The contract uses a signed message to ensure that the signer of the policy is the approved platform signer in order for a user to purchase the policy with the specified terms.
- The signed message is only valid until the "max deadline" has passed from when the message was signed. This defaults to 7 days.
- for a period of 7 days from the time it was signed.
- Users can pay for the policy in ETH or any other accepted premium currency.
- The contract uses the Exchange Agent to convert the price of the policy from USDC to the given currency, after which the tokens are sent to the Premium Pool contract.
- The contract may check each protocol requested by the user to see if it is blacklisted, if the project team has enabled this functionality in the Sales Policy Factory contract. Users may not purchase coverage for blacklisted protocols.
- The contract mints 1 ERC-721 NFT to the user for every policy purchased.
- Once a policy has expired or has been claimed, the CapitalAgent contract may burn the NFT representing the policy at any time.
- The Sales Policy Factory may pause and unpauses the contract at any time.
- The Sales Policy Factory may set the Premium Pool, Exchange Agent, Signer, and Capital Agent addresses at any time.
- The Sales Policy Factory may set the max deadline to any non-zero value at any time.
- The Sales Policy Factory may set the protocol URI at any time.
- The Sales Policy Factory may give a max approval to a specified currency address.
- The contract uses the ERC-1776 Native Meta Transactions standard to allow users to interact with the contract without paying any gas fees.

PremiumPool Contract:

- This contract is used by the project team to manage the platform's collected premium payments.

- *Whitelisted addresses may use the `collectPremiumInETH()` and `collectPremium()` functions to deposit ETH or any whitelisted tokens allocated as follows:*
 - *70% is allocated towards SSIP rewards.*
 - *20% is allocated towards the Buyback functionality.*
 - *10% is allocated towards SSRP rewards.*
- *Any address with the Admin role may use this contract to exchange the ETH and tokens collected by the contract as SSRP for USDC which is subsequently transferred to a specified Rewarder address.*
- *Any address with the Admin role may use this contract to transfer a specified amount of currency or ETH collected as SSIP rewards to a Rewarder address.*
- *Any address with the Admin role may convert a currency or ETH collected as a Buyback for UNO tokens which are subsequently transferred to the 0x...dEaD address.*
- *Any address with the Governance role may withdraw any ETH or tokens from this contract at any time*
- *Any address with the Admin role may add and remove a currency from the currency whitelist at any time.*
- *Any address with the Admin role may grant and revoke a maximum allowance for a token at any time.*
- *Any address with the Admin role may add and remove an address from the users whitelist at any time.*

ExchangeAgent Contract:

- *This contract is used throughout the protocol to facilitate token swaps as needed.*
- *Only whitelisted addresses are able to use this contract to convert tokens to ETH, ETH to tokens, or tokens to tokens when the contract is not paused.*
- *Upon swapping tokens, the user must provide an approval for the contract to use their tokens.*
- *The contract uses an Oracle Price Feed to determine the price of an asset; the Oracle was not provided to us in the scope of the audit, so we are unable to provide an assessment in terms of security.*
- *The owner may adjust the slippage used while swapping to any value at any time.*
- *The owner may add or remove any address from the whitelist at any time.*
- *The owner may pause and unpaue the contract at any time.*
- *The owner may set the Oracle Price Feed address at any time.*

EscalationManager Contract:

- *This contract is used to manage the "assertion policy" and whether disputes and assertions are allowed for a specified address.*
- *Any address with the Claim Accessor role may toggle when an address is allowed to dispute or assert.*
- *Any address with the Oracle role may use this contract to emit a PriceRequestAdded event.*

ClaimProcessor Contract:

- *This contract is used to manage and approve claims.*
- *Any address with the SSIP role can request a new claim. This will create a new claim for the specified policy ID.*
- *Any address with the Guardian Council role can approve a claim.*
- *Once a claim has been approved any address may claim the policy. This will settle the payout in the claim's associated SSIP contract.*

MultiSigWallet Contract:

- *This contract is used to execute arbitrary logic once a threshold of signers approve a transaction.*
- *Any non-zero amount of signers are set upon deployment. No signers may be added after deployment.*
- *Any signer may submit a transaction.*
- *Any signer may confirm a non-executed transaction.*
- *Each signer may only confirm a transaction once.*
- *Once the transaction reaches the required number of confirmations any address may execute the transaction.*
- *Signers may revoke their confirmation after confirming.*

AUDIT RESULTS

Vulnerability Category	Notes	Result
Arbitrary Jump/Storage Write	N/A	PASS

Centralization of Control	<ul style="list-style-type: none"> The admin role may set the SSIP and SSRP reward multiplier to any non-zero value. The Rewarder owner may withdraw reward currency from the contract. The CapitalAgent operator address may set the MCR, MLR, and SCR percentages to any non-zero value. The CapitalAgent owner may update the ExchangeAgent used to determine token value in USDC. The SalesPolicyFactory owner may blacklist a policy. The admin role may withdraw any currency from the PremiumPool. The ExchangeAgent owner may update the OraclePriceFeed address. 	WARNING
Compiler Issues	N/A	PASS
Delegate Call to Untrusted Contract	N/A	PASS
Dependence on Predictable Variables	N/A	PASS
Ether/Token Theft	N/A	PASS
Flash Loans	N/A	PASS
Front Running	N/A	PASS
Improper Events	N/A	PASS
Improper Authorization Scheme	N/A	PASS
Integer Over/Underflow	N/A	PASS
Logical Issues	N/A	PASS
Oracle Issues	N/A	PASS
Outdated Compiler Version	N/A	PASS
Race Conditions	N/A	PASS
Reentrancy	N/A	PASS
Signature Issues	N/A	PASS
Sybil Attack	N/A	PASS
Unbounded Loops	N/A	PASS
Unused Code	N/A	PASS
Overall Contract Safety		PASS

ABOUT SOURCEHAT

SourceHat (formerly Solidity Finance - founded in 2020) has quickly grown to have one of the most experienced and well-equipped smart contract auditing teams in the industry. Our team has conducted 1700+ solidity smart contract audits covering all major project types and protocols, securing a total of over \$50 billion U.S. dollars in on-chain value!

Our firm is well-reputed in the community and is trusted as a top smart contract auditing company for the review of solidity code, no matter how complex. Our team of experienced solidity smart contract auditors performs audits for tokens, NFTs, crowdsales, marketplaces, gambling games, financial protocols, and more!

Contact us today to get a free quote for a smart contract audit of your project!

WHAT IS A SOURCEHAT AUDIT?

Typically, a smart contract audit is a comprehensive review process designed to discover logical errors, security vulnerabilities, and optimization opportunities within code. A *SourceHat Audit* takes this a step further by verifying economic logic to ensure the stability of smart contracts and highlighting privileged functionality to create a report that is easy to understand for developers and community members alike.

HOW DO I INTERPRET THE FINDINGS?

Each of our Findings will be labeled with a Severity level. We always recommend the team resolve High, Medium, and Low severity findings prior to deploying the code to the mainnet. Here is a breakdown on what each Severity level means for the project:

- **High** severity indicates that the issue puts a large number of users' funds at risk and has a high probability of exploitation, or the smart contract contains serious logical issues which can prevent the code from operating as intended.
- **Medium** severity issues are those which place at least some users' funds at risk and has a medium to high probability of exploitation.
- **Low** severity issues have a relatively minor risk association; these issues have a low probability of occurring or may have a minimal impact.
- **Informational** issues pose no immediate risk, but inform the project team of opportunities for gas optimizations and following smart contract security best practices.

[G O H O M E](#)