Safe Haven Contract Audit

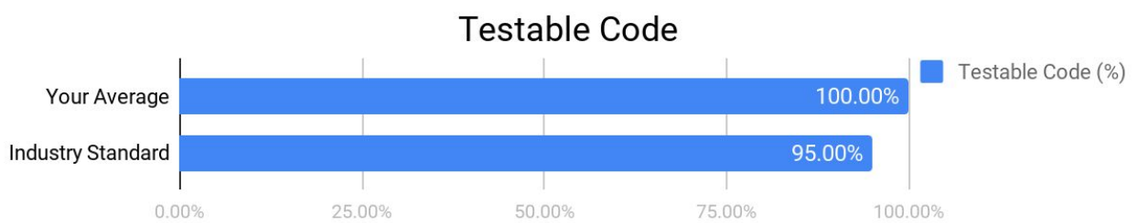
Prepared by Hosho
October 3rd, 2018

Report Version: 2.0

Executive Summary

This document outlines the overall security of Safe Haven's smart contract as evaluated by Hosho's Smart Contract auditing team. The scope of this audit was to analyze and document Safe Haven's contract codebase for quality, security, and correctness.

All issues have been resolved, and these contracts pass the rigorous auditing process performed by the Hosho team. (See Complete Analysis)



Testable code is 100% which is greater than the industry standard of 95%. (See Coverage Report)

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Hosho recommend that the Safe Haven team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table Of Contents

1. Auditing Strategy and Techniques Applied

The Hosho team has performed a thorough review of the smart contract code, the latest version as written and updated on October 1st, 2018. All main contract files were reviewed using the following tools and processes. (See All Files Covered)

Throughout the review process, care was taken to ensure that the contract:

- Documentation and code comments match logic and behavior;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks; and
- Is not affected by the latest vulnerabilities.

The Hosho team has followed best practices and industry-standard techniques to verify the implementation of Safe Haven's contract. To do so, the code is reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Meadow testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1. Due diligence in assessing the overall code quality of the codebase.
2. Cross-comparison with other, similar smart contracts by industry leaders.
3. Testing contract logic against common and uncommon attack vectors.
4. Thorough, manual review of the codebase, line-by-line.
5. Deploying the smart contract to testnet and production networks using multiple client implementations to run live tests.

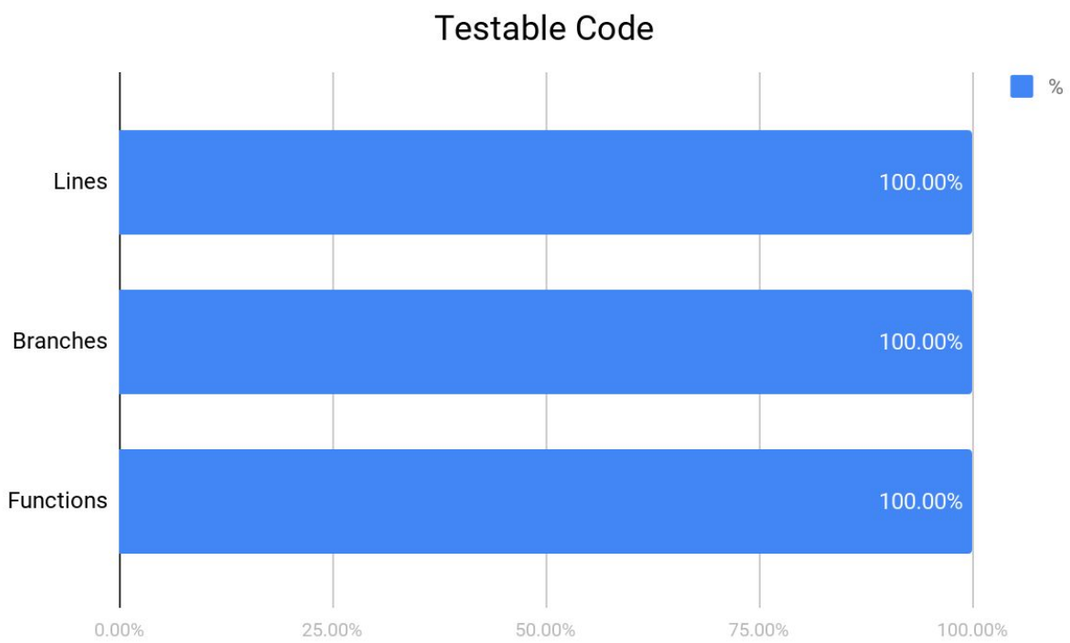
2. Structure Analysis and Test Results

2.1. Summary

The Safe Haven contracts implement a contribution pooling factory system, allowing multiple discrete pools to be initialized. Contributions to these pools are accepted and can be reversed by contributors while the contribution period is open, or if the pool cancels their funds gathering.

2.2 Coverage Report

As part of our work assisting Safe Haven in verifying the correctness of their contract code, our team was responsible for writing a unit test suite using the Meadow testing framework.



For each file see Individual File Coverage Report

2.3 Failing Tests

No failing tests.

See Test Suite Results for all tests.

3. Complete Analysis

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged "Resolved" or "Unresolved" depending on whether they have been fixed or still need addressing. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

- **Critical** - The issue affects the contract in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.
- **High** - The issue affects the ability of the contract to compile or operate in a significant way.
- **Medium** - The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.
- **Low** - The issue has minimal impact on the contract's ability to operate.
- **Informational** - The issue has no impact on the contract's ability to operate, and is meant only as additional information.

3.1 Resolved, Low: Unnecessary Branch

Contract: SplitterRepository

Explanation

The `addToken` function has an if statement `if (count > 0)`, where count is a *uint256* passed in from the Splitter contract `divideTokens` function. `divideTokens` has its own check of `if (balance > 0)` before calling `addToken`, so `divideTokens` will return *false* before the branch in `addToken` is ever able to be hit.

Resolution

The `if` statement was removed in the `addToken` function.

3.2 Informational: Unused Function

Contract: Pool

Explanation

The `maxCapNotReached` function on line 313 is not used, but there is a second function of the same name that is called from the `contribute` function.

Update

The second `maxCapNotReached` function has been removed from the contract.

3.3 Informational: Inaccurate Notation

Contract: Pool

Explanation

The `transferPoolFunds` function has the same developer notes as `transferPoolFundsWithData`, despite not taking a data argument. "Transfer the balance of this contract to the destinationAddress. Add extra data that needs to be sent to the contract (ex: a method call)”

Update

The developer notation on `transferPoolFundsWithData` has been updated with the proper information.

---

4. Closing Statement

The Hosho team is grateful to have been given the opportunity to work with the Safe Haven team.

The team of experts at Hosho, having backgrounds in all aspects of blockchain, cryptography, and cybersecurity, can say with confidence that the Safe Haven contract is free of any critical issues. Additionally, all previously noted recommendations have been applied by the Safe Haven team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

We at Hosho recommend that the Safe Haven team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

5. Appendix A

Test Suite Results

HoshoAudit.PoolTests

✓ addPool_byNonCaller_ExpectRevert (5ms)

✓ addWei_ContributerIsMsgSender_FundsRepoWithdraw (14ms)

✓ cancel_RequiresCurrentState_SetStateCancelled (27ms)

✓ cancel_StateMustBeOpen_ExpectRever (29ms)

✓ contribute_ContributionZero_FalseContributions (154ms)

✓ contribute_MaxCapReached_ExpectRevert (141ms)

✓ contribute_RequireStateOpenClosed_ExpectRevert (146ms)

✓ contribute_WeiOverPersonalCap_ExpectRevert (126ms)

✓ distributeFunds_FundsRepoAlreadyDeployed_ReturnFalse (284ms)

✓ distributeFunds_RequireStateTransferred_ExpectRevert (224ms)

✓ distributeFunds_TokenAddressNotZero_ExpectRevert (238ms)

✓ distributeFunds_WithdrawalBalanceTokensGreaterZero_ExpectRevert (241ms)

✓ distributeFunds_WithdrawTokensGreaterZero_ExpectRevert (258ms)

✓ external_call (24ms)

✓ external_call_DestinationNotFunds_ExpectRevert (26ms)

✓ external_call_DestinationNotPoolRepo_ExpectRevert (17ms)

✓ external_call_DestinationNotThis_ExpectRevert (10ms)

✓ external_call_DestinationNotToken_ExpectRevert (22ms)

✓ info_CheckInfo_ReturnPoolInfo (25ms)

✓ isThorBlock_ChecksThor_ReturnsTrue (40ms)

✓ listContributions_CurrentContributors_ReturnMap (91ms)

✓ newPool_Withdraw_EmitPoolEvent (539ms)

✓ onlyAdmin_NotAdmin_ExpectRevert (10ms)

✓ onlyOwnerModifier_notOwner_ExpectRevert (38ms)

✓ poolConstructor_DestinationNotLocked_ExpectRevert (11ms)

✓ poolConstructor_FeeAddressNot0_ExpectRevert (40ms)

✓ poolConstructor_FeeGreater1000_ExpectRevert (40ms)

✓ poolConstructor_MaxCapGreater0_ExpectRevert (40ms)

✓ poolConstructor_MaxCapGreaterMinCap_ExpectRevert (40ms)

✓ poolConstructor_MinCapGreater0_ExpectRevert (40ms)

✓ poolConstructor_PoolAddressNot0_ExpectRevert (38ms)

✓ revokeContribution_RequireAmountGreaterZero_ExpectRevert (163ms)

✓ revokeContribution_RequireStateOpenCancelled_ExpectRevert (166ms)

✓ setDestinationAddress_CantBeLocked_NewDestination (18ms)

✓ setDestinationAddress_DestinationLocked_ExpectRevert (43ms)

✓ setFeeAddress_UpdateAddress_Success (14ms)

✓ setFeePerMille_NewFee_Sucess (29ms)

✓ setTokenAddress_RequireStateOpenTransferred_ExpectRevert (180ms)

✓ setTokenAddress_StateMustBeOpen_ChangeAddress (12ms)

✓ transferPoolFunds_BadTransfer_ExpectRevert (89ms)

✓ transferPoolFundsWithData_BadTransfer_ExpectRevert (87ms)

✓ transferPoolFundsWithData_UpdateState (276ms)

✓ updateStateOnTransferPoolFunds_DesinationNotZero_ExpectRevert (39ms)

✓ updateStateOnTransferPoolFunds_RequireStateOpen_ExpectRevert (97ms)

✓ updateWhitelist_AddAndSubToWhitelist_UpdateList (52ms)

✓ updateWhitelistMultiple_AddMultipleToWhitelist_UpdateList (33ms)

✓ version_CheckVersion_Returns1 (75ms)

✓ withdrawalBalance_Withdraw_EmitPoolEvent (387ms)

HoshoAudit.SafeMathTests

✓ add_Addition_Allow (6ms)

✓ add_AdditionOverflow_ExpectRevert (2ms)

✓ div_Divide_Allow (5ms)

✓ div_DivideByZero_ExpectRevert (13ms)

✓ mul_Multiply_Allow (7ms)

✓ mul_MultiplyByZero_Allow (12ms)

✓ mul_MultiplyOverflow_ExpectRevert (26ms)

✓ sub_SubractionOverflow_ExpectRevert (9ms)

✓ sub_Subtraction_Allow (5ms)

HoshoAudit.SplitterTests

✓ divideTokens_BalanceGreaterZero_ExpectRevert (0.02s)

✓ divideTokens_FailingTransfer_ExpectRevert (0.04s)

✓ divideTokens_TransferBalance_CallsRepoAddToken (0.05s)

✓ divideWei_PayWithFallback_CallsRepoAddWei (0.04s)

✓ divideWei_WeiGreaterThanZero_ReturnFalse (0.02s)

✓ splitterRepo_ConstructorNot100_ExpectRevert (0.03s)

✓ splitterRepo_DifferentLengths_ExpectRevert (0.02s)

✓ splitterRepo_ZeroBeneficiaries_ExpectRevert (0.02s)

✓ tokenBalance_TransferTokens_ReturnRepoTokenBalance (0.03s)

✓ weiBalance_GetBalance_ReturnRepoWeiBalance (0.01s)

✓ withdraw_TransferTokens_CallsRepoWithdraw (0.03s)

✓ withdrawTokens_FailingTransfer_ExpectRevert (0.04s)

✓ withdrawTokens_TransferTokens_CallsRepoWithdrawTokens (0.04s)

6. Appendix B

All Contract Files Tested

File	Fingerprint (SHA256)
Pool.sol	2829B8671869C48C8F107DED172FDBC051649742EF2FE16815D166DE15CF2997
PoolContributionRepository.sol	760CB5E620124224BC415F3C3181D3B5433EDE25E386B3F988ACFE9AA2A4A217
PoolRepository.sol	2610B19E1C77A6E78C597BB6841D379E197C90A2E25ECAA2D6E2045A6EDD3B4B
ThorBlock.sol	D4D49F119E621F7A4E6F794A6408961C7D5A947359F29BCD5F500043249AB0E8
VersionablePool.sol	B5D918107E19599C8886D54F0567CC2E2D6D05E832C8456A9A28E0C9583CBBB8
control/Callable.sol	15D1253585FD26AFFAA14326BDC8BB66FDE839E3185924BDF67A3BAC9F1A4FCB
control/Ownable.sol	59D738F871609D83619692FA1D84D3B2608357F3CEE612879B8BB528CCE81754
math/Pct.sol	FE2766EC4881DC452FD5F0F8DFA3A9D9309F1ADBEEC5DC2538CEF905C6429BC1
math/SafeMath.sol	999CE5FDAFF7E9CEE3ED8E750F42FC0489AA637A6846CD956A56CAF642B039FB
split/Splitter.sol	F3943BB7764FE46053035E1834426B85C554FEA16D125436F17CE96A370A1D8B
split/SplitterRepository.sol	FFF29F19308AA124A1D5838AD318942B0573E89545E2C5F1CCF38346A1C47282
token/FundsRepository.sol	D5ECF07056ED4927399F9E775AB8F316DCFED084275AA4FB9A5598DB3FE88076
token/erc20/ERC20.sol	3599A168AD93C764C795047B135F34520C41BF7E5496259BB19E733460CC99C0
token/erc20/SimpleToken.sol	66BB3B88043DC4DF3606EB34099F41AAA4C8BB0651D433E8667E78F8E539CDCC
token/erc20/StandardToken.sol	B2EB1B13AD734B90B7A9072B1C340393CAE74E682A0616D53B7CC2E3EC7446DD
token/erc20/StrippedERC20.sol	4D9653C9B8E2CF505914BDE143DF984A3D0F388F0165942F69161F9FEE379764

7. Appendix C

Individual File Coverage Report

File	% Lines	% Branches	% Functions
Pool.sol	100.00%	100.00%	100.00%
PoolContributionRepository.sol	100.00%	100.00%	100.00%
PoolRepository.sol	100.00%	100.00%	100.00%
ThorBlock.sol	100.00%	100.00%	100.00%
VersionablePool.sol	100.00%	100.00%	100.00%
control/Callable.sol	100.00%	100.00%	100.00%
control/Ownable.sol	100.00%	100.00%	100.00%
math/Pct.sol	100.00%	100.00%	100.00%
math/SafeMath.sol	100.00%	100.00%	100.00%
split/Splitter.sol	100.00%	100.00%	100.00%
split/SplitterRepository.sol	100.00%	100.00%	100.00%
token/FundsRepository.sol	100.00%	100.00%	100.00%
token/erc20/ERC20.sol	100.00%	100.00%	100.00%
token/erc20/SimpleToken.sol	100.00%	100.00%	100.00%
token/erc20/StandardToken.sol	100.00%	100.00%	100.00%
token/erc20/StrippedERC20.sol	100.00%	100.00%	100.00%
All files	**100.00%**	**100.00%**	**100.00%**