Quantstamp Contract Security Certificate

January 9th 2020 — Quantstamp Verified

OmiseGO Plasma Framework

This smart contract audit was prepared by Quantstamp, the protocol for securing smart contracts.

QUANTSTAMP VERIFIED SECURITY CERTIFICATE

Executive Summary

Type	Protocol
Auditors	Kacper Bąk, Senior Research Engineer Ed Zulkoski, Senior Security Engineer Jan Gorzny, Blockchain Researcher
Timeline	2019-10-21 through 2020-01-10
EVM	Byzantium
Languages	Solidity, Javascript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	More Viable Plasma Integration Doc Architecture and Design

Source Code

Repository	Commit
plasma-contracts	bab16ae

Goals

- Can a malicious user compromise funds from OMG network users?
- Can a malicious user disrupt users to deposit, withdraw or perform transactions on the OMG network?
- Can users safely exit their funds in the presence of malicious owner?
- How can the existing code be improved?

Changelog

- 2019-11-22 - Initial report
- 2020-01-10 - Reaudit based on commit 9d79e35

Overall Assessment	We have done a thorough review of OmiseGO's implementation of Plasma Framework. The audit included the design, architecture, and the actual codebase. Although most of the issues we have found are minor, we consider the denial-of-service issue (already fixed) a high-severity one, whereas UTXO fragmentation as a medium-severity one. Overall, the code conforms to the presented specification. It is mostly well-documented and adheres to best practices. It features good modularization and architecture. In the report, we provide further ideas for design and code improvements. Design-wise, we think that piggybacking on IFE introduces unnecessary complexity for end-users, and, therefore, we encourage carrying out additional research to smoothen out end-user experience. **Update:** we reaudited the code based on commit `9d79e35`. Since the diff between this and the previous commit `bab16ae` contains over 250 commits and significant code changes, our reaudit was only concerned with the changes that addressed findings from the initial report.

Total Issues	8	(6 Resolved)
High Risk Issues	1	(1 Resolved)
Medium Risk Issues	1	(0 Resolved)
Low Risk Issues	5	(4 Resolved)
Informational Risk Issues	0	(0 Resolved)
Undetermined Risk Issues	1	(1 Resolved)

1 Unresolved
1 Acknowledged
6 Resolved

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
? Undetermined	The impact of the issue is uncertain.

Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	the issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Resolved	Adjusted program implementation, requirements or constraints to eliminate the risk.

Summary of Findings

ID	Description	Severity	Status
QSP-1	Denial-of-Service (DoS)	High	Resolved
QSP-2	UTXO fragmentation may make exits prohibitively expensive	Medium	Acknowledged
QSP-3	Merkle tree susceptible to second preimage attack	Low	Resolved
QSP-4	Possible out of bounds access in `RLPReader._itemLength()`	Low	Resolved
QSP-5	Race Conditions / Front-Running	Low	Unresolved
QSP-6	Enforce ownership renouncement	Low	Resolved
QSP-7	A malicious maintainer may steal funds by setting a bad deposit verifier	Low	Resolved
QSP-8	Maintainer may register a malicious game	? Undetermined	Resolved

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Truffle
- Ganache
- SolidityCoverage
- Mythril
- Securify
- Slither

Steps taken to run the tools:

1. Installed Truffle: `npm install -g truffle`
2. Installed Ganache: `npm install -g ganache-cli`
3. Installed the solidity-coverage tool (within the project's root directory): `npm install --save-dev solidity-coverage`
4. Ran the coverage tool from the project's root directory: `./node_modules/.bin/solidity-coverage`
5. Installed the Mythril tool from Pypi: `pip3 install mythril`
6. Ran the Mythril tool on each contract: `myth -x path/to/contract`
7. Ran the Securify tool: `java -Xmx6048m -jar securify-0.1.jar -fs contract.sol`
8. Installed the Slither tool: `pip install slither-analyzer`
9. Run Slither from the project directory `slither .`

Assessment

Findings

QSP-1 Denial-of-Service (DoS)

Severity: *High Risk*

Status: Resolved

File(s) affected: `ExitGameController.sol`, `EthVault.sol`, `PaymentProcessStandardExit.sol`, `PaymentChallengeIFEInputSpent.sol`, `PaymentChallengeIFEOutputSpent.sol`, `PaymentProcessInFlightExit.sol`

Description: A Denial-of-Service (DoS) attack is a situation in which an attacker renders a smart contract unusable. In the context of `ExitGameController.processExits()`, the line 49 of `EthVault.sol` may get executed to return ETH during an exit. Since there is no restriction on how much gas `receiver` will get, a malicious receiver may decide to use up all the forwarded gas which may lead to reverting the transaction that invoked `ExitGameController.processExits()`. Consequently, (for a given token and vault), one may block all of the subsequent exits. Similar attack is possible for malicious ERC20 tokens (via `Erc20Vault.sol#53`).
Additionally, please note the following lines that forward all gas to the receiver:

- `PaymentProcessStandardExit.sol#58`,
- `PaymentChallengeIFEInputSpent.sol#114`,
- `PaymentChallengeIFEOutputSpent.sol#70`, and
- `PaymentProcessInFlightExit.sol#92` and `#108`

Exploit Scenario: See Issue 385.

Recommendation: For ETH transfers, we recommend defining a custom `transfer()` method that performs the transfer with a predefined gas limit. The custom `transfer()` function shall be used in place of `call.value()` without the gas limit. The issue has been addressed in PR 248.
For ERC20 tokens we recommend that the maintainer carefully reviews their code to ensure that the DoS/gas griefing attack does not occur.

QSP-2 UTXO fragmentation may make exits prohibitively expensive

Severity: *Medium Risk*

Status: Acknowledged

Description: The support for micro-transactions is an important use case for Plasma. Micro-transaction may lead to UTXO fragmentation. Consequently, if a user (or set of users) is in possession of a significant number of small valued UTXOs, it may be prohibitively expensive to exit the UTXOs from the Plasma chain due to the gas cost. Furthermore, malicious operator could take advantage of such a situation and steal funds while performing an exit. Although, UTXO fragmentation is a known Plasma issue, the audited codebase does not address this problem directly. See Issue 427.

Exploit Scenario:

1. The operator notices that a user (or several users) have many low value UTXOs, such that it is not gas-efficient to exit them.
2. The operator spawns an out-of-nowhere transaction to exit, which has value proportional to the sum of these low valued UTXOs. (The value of operator's transaction would likely have to be somewhat lower than the full sum, in case a few of the low valued UTXOs exit anyway.)
3. The users decide to not exit since it would be too costly, and the operator is able to exit the funds.

Recommendation: The issue could largely be prevented if users/wallets regularly and proactively consolidate their funds on the Plasma chain by creating transactions that send multiple input UTXOs to themselves with a single output. It is important to note that consolidation needs to happen before an attack occurs, or else the operator would have higher priority when exiting the funds.

QSP-3 Merkle tree susceptible to second preimage attack

Severity: Low Risk

Status: Resolved

File(s) affected: `Merkle.sol`

Description: Second preimage attack is a well-known issue with Merkle tree implementations. In the current implementation, the function `Merkle.checkMembership()`may be called for intermediate nodes as opposed to checking only leaves.

Exploit Scenario: See Issue 425.

Recommendation: We recommend prepending a byte to each node in the tree to discriminate between leaf and intermediate nodes. Second part of the an effective solution is to fix the size of the Merkle tree. As the team informed us, the height of the used Merkle trees is always 16.

QSP-4 Possible out of bounds access in `RLPReader._itemLength()`

Severity: Low Risk

Status: Resolved

File(s) affected: `RLPReader.sol`

Related Issue(s): SWC-110

Description: The function `RLPReader._itemLength()` takes a `memPtr` and attempts to decode an RLP encoded item to return its complete size (tag + length + payload). The function does not have any information about the length of the buffer it is working on. Consequently, it cannot check if it reads out-of-bounds. It has to rely on the caller to do bounds checking.

Exploit Scenario: See Issue 409.

Recommendation: We recommend passing the buffer (instead of the memory pointer) and performing bounds checking while decoding. The issue has been addressed in PR 396.

QSP-5 Race Conditions / Front-Running

Severity: Low Risk

Status: Unresolved

File(s) affected: `PaymentStandardExitRouter.sol`, `PaymentInFlightExitRouter.sol`

Related Issue(s): SWC-114

Description: A block is an ordered collection of transactions from all around the network. It's possible for the ordering of these transactions to manipulate the end result of a block. A miner attacker can take advantage of this by generating and moving transactions in a way that benefits themselves.
Specifically, as anyone can see the challenge transactions in the transaction pool, they may try to front-run them. This may disincentivize legitimate users from attempting challenges. The following functions are vulnerable to front-running:

- `PaymentStandardExitRouter.challengeStandardExit()`,
- `PaymentInFlightExitRouter.challengeInFlightExitOutputSpent()`,
- `PaymentInFlightExitRouter.challengeInFlightExitInputSpent()`, and
- `PaymentInFlightExitRouter.challengeInFlightExitNotCanonical()`

Recommendation: Currently we have no recommendation. One could design a protocol that is resistant to TOD, but it would make the implementation of challenges more complex. At this point, the benefit of doing that is unclear.

QSP-6 Enforce ownership renouncement

Severity: Low Risk

Status: Resolved

File(s) affected: `OutputGuardHandlerRegistry.sol`, `SpendingConditionRegistry.sol`

Description: Both `OutputGuardHandlerRegistry` and `SpendingConditionRegistry` are expected to have their ownership renounced before injecting the registries to the `ExitGame` contracts. Although stated in the documentation, this requirement is not enforced programmatically. Hence, there is nothing requiring the owner to actually renounce the ownership. As such, the owner could register new output guard handlers and spending conditions to perform malicious operations on the Plasma chain.

Recommendation: We recommend enforcing the ownership renouncement programmatically. For example, the vaults may reject any deposits as long as `OutputGuardHandlerRegistry` and `SpendingConditionRegistry` have owners.

QSP-7 A malicious maintainer may steal funds by setting a bad deposit verifier

Severity: *Low Risk*

Status: Resolved

File(s) affected: `Vault.sol`

Description: If a malicious maintainer (colluding with the operator) sets a faulty deposit verifier, they may be able to submit a fake deposit and exit funds before other standard exits. It is possible because: 1) the function `Vault.setDepositVerifier()` sets the maturity timestamp (on line 52) to be a single exit period, and 2) exits of deposits are shorter than standard exits (as defined in `ExitableTimestamp.sol`).

Exploit Scenario:

1. At time `T_0`, the maintainer sees a very large payment of `N` tokens from Alice to Bob, which is still in-flight. At this moment, the maintainer first sets a new deposit verifier, which will allow the maintainer to verify any deposit they want. This will only go into effect after the delay as specified on `Vault.sol#L52` (i.e., after 1 exit period). Shortly after, the in-flight transaction is included in the child chain, say at time `T_0 + delta`.
2. At some point soon after, Bob realizes that this new deposit verifier is fraudulent. Since Bob sees the transaction included in the child chain, he attempts a *standard* exit on the transaction. This exit will be allowed to complete at time `T_0 + delta + 2 exit periods`.
3. At time `T_0 + 1 exit period`, the malicious maintainer creates a fake deposit indicating that they've deposited `N` tokens, where `N` is the amount in the payment above. This will be scheduled for exit at time `T_0 + 2 exit periods`.
4. The malicious maintainer exits at time `T_0 + 2 exit periods`.
5. Bob cannot exit at time `T_0 + delta + 2 exit periods`, since in step (4) all the funds were wiped.

Recommendation: We have the following recommendations. First, users should be made aware of scenarios where standard exits are insufficient. Second, enhance the watcher so that it can recognize a new deposit verifier, informs the user, and, if necessary, performs an in-flight exit of user's funds (even if the transaction is included on the side chain). Furthermore, consider a longer delay before the new deposit verifier goes into effect so that the user is able to start a standard exit to recover their funds.

QSP-8 Maintainer may register a malicious game

Severity: *Undetermined*

Status: Resolved

File(s) affected: `ExitGameRegistry.sol`

Description: Maintainer has the privilege of adding new exit games. Each new exit game is quarantined for three exit periods -- that's the period in which the community of Plasma chain users may audit the newly added exit game. In case they realize that the game is malicious, they may decide to exit the chain. Given that it may take two exit periods to perform an exit, is it realistic that the community discovers a problematic exit game within a single exit period?

Recommendation: Currently we have no recommendation.

Automated Analyses

Mythril

Mythril reported the following:

- multiple Calls in a Single Transaction in `EthVault.withdraw(address,uint256)` and `Erc20Vault.sol#52`,
- Exception State in `EthVault.sol#34`, `EthVault.sol#49`, `Vault.sol#21`, `PriorityQueue.sol#75`, `Erc20Vault.sol#30`, and `EthVault.sol#49`,
- External Call To User-Supplied Address in `EthVault.sol#49`,
- Unchecked Call Return Value in `EthVault.sol#49`,
- Unprotected Ether Withdrawal in `EthVault.sol#49`,
- Integer Overflow in `Erc20Vault.sol`,
- External Call To Fixed Address in `Erc20Vault.sol#57`, and
- External Call To User-Supplied Address in `Erc20Vault.sol#57`.

Upon closer inspection, we classified them as non-issues.

Securify

Securify did not report any issues.

Slither

Slither reported the following:

- ETH transfer to arbitrary user in: `EthVault.sol#49`, `PaymentProcessStandardExit.sol#58`, `PaymentChallengeIFEInputSpent.sol#114`, `PaymentChallengeStandardExit.sol#96`, and `PaymentChallengeIFEOutputSpent.sol#70`.
- the following functions declared views that contain assembly code: `Address.isContract()`, `Merkle.checkMembership()`, `ECDSA.recover()`, `RLPReader.toRlpItem()`, `RLPReader.isList()`, `RLPReader.toRlpBytes()`, `RLPReader.toUint()`, `RLPReader._itemLength()`, `RLPReader._payloadOffset()`, `RLPReader.copy()`, and `RLPReader.toBytes()`.
- the following local variables that are never initialized: `PaymentProcessInFlightExit.sol#92`, `PaymentProcessInFlightExit.sol#108`, `PaymentProcessInFlightExit.sol#126`, `PaymentProcessInFlightExit.sol#199`, `PaymentProcessInFlightExit.sol#207`, `ZeroHashesProvider.sol#9`, and `PaymentStartInFlightExit.sol#142`.
- return values by external calls are ignored in `PaymentStartStandardExit.sol#214`, `PaymentPiggybackInFlightExit.sol#180`, and `ExitGameController.sol#165`.

Upon closer inspection, we classified them as non-issues.

Adherence to Specification

The specification is comprehensive and well-written. We have, however, the following concerns:

- the functions `EthDepositVerifier.verify()` and `Erc20DepositVerifier.verify()` don't check `decodedTx.outputs[0].outputType`. **Update:** resolved.

- from the design standpoint, it is unclear why users must issue a second transaction to piggyback an IFE that they created. Furthermore, it is unclear what the use case for initiating an in-flight exit by non-participant is.

Code Documentation

The code is mostly well-documented, however, in the comment block for `PaymentInFlightExitRouterArgs.ChallengeCanonicityArgs`, field descriptions are missing for `inputTx` and `inputUtxoPos`. **Update:** resolved.

Adherence to Best Practices

Although the code uses the experimental feature `ABIEncoderV2`, overall, it conforms to best practices and it makes good use of modularization. We provide further ideas for improvements below:

- in `FailFastReentrancyGuard.sol` there is no need to import `PlasmaFramework.sol`. We recommend importing `ExitGameController.sol` instead. **Update:** resolved.

- in `Erc20DepositVerifier.sol#46` there is a strict equality requiring user to approve an exact amount of tokens. Such a design prohibits the user from making a single approval for multiple deposits. To facilitate flexibility and to reduce gas usage, we recommend using `>=` instead of equality. Furthermore, the comment in line `Erc20Vault.sol#32` would hint on the recommended approach. **Update:** resolved.

- in `PaymentInFlightExitModelUtils.isFirstPiggybackOfTheToken()` there is an unused variable `isPiggybackInput` which is write-only. We recommend removing it. **Update:** resolved.

- in `WireTransaction.sol` following the line 32, we recommend checking the length of `output`, e.g., `require(4 == output.length, "Invalid output encoding")`. **Update:** resolved.

- in `VaultRegistry.sol` the function `registerVault()` takes `uint256 _vaultId` as an argument. We recommend simplifying the code by introducing a unique internal counter that could then be used to generate `_vaultId` and return it from the function.

- `utxoPos` has inconsistent definitions; for example, it is defined as `uint192` in `PaymentExitDataModel.StandardExit` and `PaymentStandardExitRouterArgs.StartStandardExitArgs`, whereas it is defined as `uint256` in `ExitChallenged.PaymentStandardExitRouter` and `UtxoPosLib.UtxoPos`. **Update:** resolved.

- similarly, `exitId` is defined as `uint160` in `ExitId.getStandardExitId()`, whereas is it defined as `uint192` in `PaymentExitDataModel.StandardExitMap`. **Update:** resolved.

- unlocked pragma in `RLPReader.sol`. Since it's a clone-and-owned library, we recommend locking pragma as in other contracts. **Update:** resolved.

- since the max number of outputs and inputs associated with a UTXO is 4, the data type for indices could be `uint8` instead of `uint16` (e.g., the return value of `UtxoPosLib.outputIndex()`, or input of `PaymentOutputToPaymentTxCondition.verify()`).

- it is unclear why the function `PaymentExitGame.processExit()` takes a second unnamed argument (presumably vault Id). If it is not used, we recommend removing it.

- it is unclear why `OutputId.computeDepositOutputId()`takes `_outputIndex` as an argument. If deposit's output UTXO is always zero, we recommend removing it.

- in `PaymentInFlightExitModelUtils.sol`, it is unclear why there are both `clearOutputPiggyback()` and `clearOutputPiggybacked()`. The former appears wrong since it does not shift the `index`. However, is not used anywhere in the code base; we recommend removing it. **Update:** resolved.

- typos: in `EthVault.sol`: "unucessful" -> "unsuccessful", and `ExitId.sol`: "depost" -> "deposit". **Update:** resolved.

- constants `MAX_INPUT_NUM` and `MAX_OUTPUT_NUM` are re-defined in multiple different places (e.g., `PaymentInFlightExitModelUtils.sol`, `PaymentExitDataModel.sol`, `PaymentPiggybackInFlightExit.sol`, `PaymentTransactionModel.sol`, `PaymentEip712Lib.sol`). We recommend extracting them to some shared contract/library. **Update:** resolved.

- the field `EthVault.withdrawEntryCounter` is not used anywhere in the code. We recommend removing it. **Update:** resolved.

Test Results

Test Suite Results

```
Contract: PlasmaFramework - Fee Claim
  Given contracts deployed, ETH exitQueue added to the framework
    When Alice deposits ETH to the plasma
      When Alice transfers to Bob, fee is implicitly collected
        And then operator mined the block with the first fee tx claiming the fee
          When Alice transfers to Carol, fee is implicitly collected
            And then operator mined the block with the second fee tx claiming the fee
              And then the operator spends the 2 fee claim outputs in a payment transaction
                 ✓ should be able to standard exit the fee via payment transaction (631ms)
                 ✓ should be able to in-flight exit the fee via Payment transaction (2865ms)

Contract: PaymentExitGame - In-flight Exit - End to End Tests
  Given contracts deployed, exit game and both ETH and ERC20 vault registered
    >>> TEST CASE: Canonical Basic
      Given Alice deposited ETH
```

```
        Given Alice started a canonical in-flight exit from transaction to Bob that is not mined
          And owner of the output (Bob) piggybacks
            ✓ should emit InFlightExitOutputPiggybacked event
            And someone processes exits for ETH after two weeks
              ✓ should transfer the funds to the output owner (Bob)
              ✓ should publish an event
              ✓ should mark output as spent
    >>> TEST CASE: Non-Canonical Basic
      Given Alice deposited ETH two times, creating output A and output B
        When Alice signs a tx1 to Bob using output A and output B as input
          And then Alice also signs another competing tx2 to Carol using output A as input
            When Bob starts IFE on tx1
              And Bob piggybacks the output of tx1
                Then the IFE of tx1 is challenged non-canonical by Carol
                  And then Alice piggybacks both outputs output A and output B
                    And then the piggyback of output A is challenged with tx2
                      When someone processes exits after two weeks
                        ✓ should return only funds of output B to Alice (input exited)
                        ✓ should NOT return fund to Bob (output not exited)
    >>> TEST CASE: Competing IFEs With One Canonical And The Other Non Canonical
      Given Alice and Bob deposits, creating output A and output B respectively
        Given Alice and Bob both sign the tx1 spending output A and output B as input
          And then Bob signs another tx2 spending output B
            Bob starts IFE for tx2 first
              Nex Alice starts IFE for tx1
                Both IFEs are challenged non-canonical
                  Suddenly Operator (Authority) includes tx2 and respond to the IFE of tx2
                    Alice piggybacks the input of tx1 and Bob piggybacks the output of tx2
                      Someone processes both exits after 2 weeks
                        ✓ should exit the output of tx2 to Bob
                        ✓ should return IFE bond for tx2 to IFE responder (Operator)
                        ✓ should exit the input of tx1 to Alice

  Contract: PaymentExitGame - Standard Exit - End to End Tests
    Given contracts deployed, exit game and both ETH and ERC20 vault registered
      ✓ should not allow to call processExit from outside of exit game controller contract (89ms)
      Given Alice deposited with ETH
        ✓ should have transferred the ETH from Alice to vault
        When Alice starts standard exit on the deposit tx
          ✓ should save the StandardExit data when successfully done (54ms)
          ✓ should put the exit data into the queue of framework (119ms)
          And then someone processes the exits for ETH after a week
            ✓ should return the fund plus standard exit bond to Alice
      Given Alice deposited ETH and transferred some to Bob
        When Bob tries to start the standard exit on the transfered tx
          ✓ should start successfully (47ms)
          And then someone processes the exits for ETH after two weeks
            ✓ should return the output amount plus standard exit bond to Bob
      Given Alice deposited ETH and transferred some to Bob
        When Alice tries to start the standard exit on the deposit tx
          ✓ should still be able to start standard exit even already spent
          Then Bob can challenge the standard exit spent
            ✓ should challenge it successfully
            ✓ should transfer the bond to Bob
            ✓ should not allow Alice to restart the exit (978ms)
            And then someone processes the exits for ETH after two weeks
              ✓ should be omitted
              ✓ should not withdraw funds from the vault
      Given Alice deposited with ERC20 token
        ✓ should have transferred the ERC20 token from Alice to vault
        Given ERC20 token added to the PlasmaFramework
          ✓ should have the ERC20 token
          When Alice starts standard exit on the ERC20 deposit tx
            ✓ should start successfully (55ms)
            And then someone processes the exits for the ERC20 token after a week
              ✓ should return the standard exit bond in ETH to Alice
              ✓ should return ERC20 token with deposited amount to Alice

  Contract: StandardExit getter Load Test
    Given contracts deployed, exit game and ETH vault registered
      Given Alice deposited with ETH
        When Alice starts standard exits on the deposit txs
          ✓ should save the StandardExit data when successfully done (30155ms)

  Contract: PriorityQueue Load Test
    load test on gas limit
      - should test and print the gas cost for each batch size items

  Contract: FeeClaimOutputToPaymentTxCondition
    verify
      ✓ should fail when fee claim output index is not 0
      ✓ should fail when fee tx is not with the expected tx type (252ms)
      ✓ should fail when fee claim output is not with the expected output type (208ms)
      ✓ should fail when payment tx is not with the expected tx type (386ms)
      ✓ should fail when payment tx does not point to the fee claim output (349ms)
      ✓ should fail when failed to recover the signer from the signature (401ms)
      ✓ should fail when payment tx not correctly signed by the fee claim output owner (410ms)
      ✓ should return true when all verification passes (403ms)

  Contract: PaymentExitGame
    ✓ should fail to deploy if the spending condition registry has not renounced its ownership (613ms)

  Contract: PaymentExitGame - Reentrant Protected
    standard exit functions are protected
      ✓ should not be able to re-enter startStandardExit (78ms)
      ✓ should not be able to re-enter challengeStandardExit (87ms)
    in-flight exit functions are protected
      ✓ should not be able to re-enter startInFlightExit (82ms)
      ✓ should not be able to re-enter piggybackInFlightExitOnInput (81ms)
      ✓ should not be able to re-enter piggybackInFlightExitOnOutput (94ms)
      ✓ should not be able to re-enter challengeInFlightExitNotCanonical (152ms)
      ✓ should not be able to re-enter respondToNonCanonicalChallenge (122ms)
      ✓ should not be able to re-enter challengeInFlightExitInputSpent (137ms)
      ✓ should not be able to re-enter challengeInFlightExitOutputSpent (167ms)
      ✓ should not be able to re-enter deleteNonPiggybackedInFlightExit (165ms)

  Contract: PaymentInFlightExitModelUtils
    isInputEmpty
      ✓ returns true when the input is empty (233ms)
      ✓ returns false when the outputId is not empty (348ms)
      ✓ returns false when the exitTarget is not empty (398ms)
      ✓ returns false when the token is not empty (393ms)
      ✓ returns false when the amount is not empty (409ms)
      ✓ returns false when the piggyback bond is not empty (400ms)
  isOutputEmpty
     ✓ returns true when the output is empty (184ms)
     ✓ returns false when the outputId is not empty (405ms)
     ✓ returns false when the exitTarget is not empty (406ms)
     ✓ returns false when the token is not empty (397ms)
     ✓ returns false when the amount is not empty (378ms)
     ✓ returns false when the piggyback bond is not empty (382ms)
  isInputPiggybacked
     ✓ returns true for piggybacked input (178ms)
     ✓ returns false for non-piggybacked input (174ms)
     ✓ fails for invalid input index (73ms)
  isOutputPiggybacked
     ✓ returns true for piggybacked output (174ms)
     ✓ returns false for non-piggybacked output (214ms)
     ✓ fails for invalid output index (42ms)
  setInputPiggybacked
     ✓ sets input as piggybacked (312ms)
     ✓ does not unset already piggybacked input (316ms)
     ✓ fails for invalid input index (38ms)
  setOutputPiggybacked
     ✓ sets outputs as piggybacked (299ms)
     ✓ does not unset already piggybacked output (314ms)
     ✓ fails for invalid output index (39ms)
  clearInputPiggybacked
     ✓ clears piggybacked input (281ms)
     ✓ does not change not piggybacked input (320ms)
     ✓ fails for invalid input index (40ms)
  clearOutputPiggybacked
     ✓ clears piggybacked output (304ms)
     ✓ does not change not piggybacked output (300ms)
     ✓ fails for invalid output index (40ms)
  isInFirstPhase
     ✓ returns true when ife is in the first phase
     ✓ returns false when ife is not in the first phase (43ms)
     ✓ returns false when ife just passes the first phase (MIN_EXIT_PERIOD/2) (39ms)

Contract: PaymentInFlightExitRouter
   ✓ should fail when being deployed with unset ETH vault (132ms)
   ✓ should fail when being deployed with unset ERC20 vault (134ms)

Contract: PaymentStandardExitRouter
   ✓ should fail when being deployed with unset ETH vault (107ms)
   ✓ should fail when being deployed with unset ERC20 vault (120ms)

Contract: PaymentTransactionStateTransitionVerifier
  verifies state transition
     ✓ should return true for a valid state transition (515ms)
     ✓ should verify transition for any combination of inputs / outputs numbers (5704ms)
     ✓ should return false when in-flight transaction overspends (590ms)
     ✓ should return false when input transactions list and utxos positions differ in length
     ✓ should revert when sum of outputs overflows uint256 (545ms)

Contract: PaymentExitGame - Update Bond
  updateStartStandardExitBondSize
     ✓ should emit an event when the standard exit bond size is updated
     ✓ should update the bond value after the waiting period has passed
  updateStartIFEBondSize
     ✓ should emit an event when the in-flight exit bond size is updated
     ✓ should update the bond value after the waiting period has passed
  updatePiggybackBondSize
     ✓ should emit an event when the in-flight exit bond size is updated
     ✓ should update the bond value after the waiting period has passed

Contract: PaymentChallengeIFEInputSpent
  challenge in-flight exit input spent
     ✓ should fail when paying out piggyback bond fails (613ms)
    after successfully challenged IFE input spent
       ✓ should emit InFlightExitInputBlocked event
       ✓ should remove the input from piggybacked (91ms)
       ✓ should pay the piggyback bond to the challenger
    check exitMap before and after challenge
       ✓ should remove the input from piggybacked (745ms)
    failures
       ✓ should fail when ife not started (170ms)
       ✓ should fail when the indexed input has not been piggybacked (205ms)
       ✓ should fail when challenging tx is the same as in-flight one (217ms)
       ✓ should fail when the challenging transaction input index is incorrect (1139ms)
       ✓ should fail when challenging tx is not of MoreVP protocol (620ms)
       ✓ should fail when the spent input is not the same as piggybacked input (331ms)
       ✓ should fail when spending condition for given output is not registered (937ms)
       ✓ should fail when spending condition is not met (1180ms)

Contract: PaymentChallengeIFENotCanonical
  challenge in-flight exit non-canonical
     ✓ should successfully challenge when transaction that created input tx is a deposit (451ms)
  challenge in-flight exit non-canonical
    when successfully challenge inFlight exit
       ✓ should emit InFlightExitChallenged event
       ✓ should set the oldest competitorPosition (92ms)
       ✓ should set the bond owner to challenger (92ms)
       ✓ should flag the exit non-canonical (85ms)
    is unsuccessful and
       ✓ fails when provided input index is bigger then number of in-flight transaction inputs (181ms)
       ✓ fails when competing tx is the same as in-flight one (284ms)
       ✓ fails when provided input tx does not match input tx stored in in-flight exit data (294ms)
       ✓ fails when first phase is over (286ms)
       ✓ fails when competing tx is not included in the given position (846ms)
       ✓ fails when ife not started (182ms)
       ✓ fails when spending condition is not met (592ms)
       ✓ fails when spending condition for given output is not registered (666ms)
       ✓ fails when competing tx is younger than already known competitor (1416ms)
       ✓ fails when challenge with the same competing tx twice (1291ms)
       ✓ should set large competitor position when competitor is in-flight (427ms)
  challenge in-flight exit non-canonical with non MoreVP tx
     ✓ fails when competing tx position is not in a block (783ms)
     ✓ fails when competing tx is included in a block (846ms)
  response to non-canonical challenge
    when successfully responded to non-canonical challenge
       ✓ should emit InFlightExitChallengeResponded event (122ms)
       ✓ should set isCanonical back to true (216ms)
       ✓ should set bond owner to caller (225ms)
       ✓ should set oldest competitor position to response position (203ms)
    is unsuccessful and
       ✓ fails when in-flight exit does not exists (119ms)
```

```
        ✓ fails when in-flight transaction is not older than competitor (110ms)
        ✓ fails when in-flight transaction position is 0 (110ms)
        ✓ fails when the block of the in-flight tx position does not exist (135ms)
        ✓ fails when in-flight transaction is not included in block (171ms)

  Contract: PaymentChallengeIFEOutputSpent
    challengeInFlightExitOutputSpent
      ✓ should emit event when challenge is successful (681ms)
      ✓ should pay out bond to the challenger when challenged successfully (685ms)
      ✓ should fail when paying out piggyback bond fails (707ms)
      ✓ should not allow to challenge output exit successfully for the second time (989ms)
      ✓ should fail when exit is not started (160ms)
      ✓ should fail when output is not piggybacked (215ms)
      ✓ should fail when challenging tx is not of MoreVP protocol (856ms)
      ✓ should fail when in-flight transaction is not included in Plasma (749ms)
      ✓ should fail when spending condition for challenging tx is not registered (1343ms)
      ✓ should fail when challenging transaction does not spend the output (1505ms)

  Contract: PaymentChallengeStandardExit
    challengeStandardExit
      When spending condition not registered
        ✓ should fail by not able to find the spending condition contract (1074ms)
      Given everything registered
        ✓ should fail when malicious user tries attack when paying out bond (676ms)
        ✓ should fail when exit for such exit id does not exist (368ms)
        ✓ should fail when try to challenge with a tx that is not of MoreVP protocol (860ms)
        ✓ should fail when spending condition contract returns false (1141ms)
        ✓ should fail when spending condition contract reverts (1127ms)
        ✓ should fail when provided exiting transaction does not match stored exiting transaction (1082ms)
        ✓ should call the Spending Condition contract with expected params (733ms)
        When successfully challenged
          ✓ should set exitable flag to false when successfully challenged
          ✓ should transfer the standard exit bond to challenger when successfully challenged
          ✓ should emit ExitChallenged event when successfully challenged

  Contract: PaymentDeleteInFlightExit
    deleteNonPiggybackedInFlightExit
      ✓ should fail when the exit does not exits (124ms)
      ✓ should fail when the exit is still in first phase (313ms)
      ✓ should fail when input of the exit is piggybacked (334ms)
      ✓ should fail when output of the exit is piggybacked (355ms)
      ✓ should fail when failed to transfer bond (382ms)
      when the delete succeeds
        ✓ should send the IFE bond to the bond owner
        ✓ should delete the in-flight exit data (104ms)
        ✓ should emit InFlightExitDeleted event

  Contract: PaymentPiggybackInFlightExitOnInput
    piggybackOnInput
      ✓ should fail when not send with the bond value (86ms)
      ✓ should fail when no exit to piggyback on (268ms)
      ✓ should fail when first phase of exit has passed (317ms)
      ✓ should fail when input index exceed max size of tx input (378ms)
      ✓ should fail when the input is empty (372ms)
      ✓ should fail when the same input has been piggybacked (456ms)
      ✓ should fail when not called by the exit target of the output (464ms)
      ✓ should fail when there is no block for the exit position to enqueue (496ms)
      When piggyback successfully
        ✓ should enqueue with correct data when it is the first piggyback of the exit on the token
        ✓ should not enqueue when it is not first piggyback of the exit on the same token (234ms)
        ✓ should set the exit as piggybacked on the input index (84ms)
        ✓ should set a proper piggyback bond size (99ms)
        ✓ should emit InFlightExitInputPiggybacked event
      When piggyback successfully on ERC20 input
        ✓ should enqueue with correct data when it is the first piggyback of the exit on the token

  Contract: PaymentPiggybackInFlightExitOnOutput
    piggybackOnOutput
      ✓ should fail when not send with the bond value (74ms)
      ✓ should fail when no exit to piggyback on (148ms)
      ✓ should fail when first phase of exit has passed (331ms)
      ✓ should fail when output index exceed max size of tx output (381ms)
      ✓ should fail when the indexed output is empty (391ms)
      ✓ should fail when the same output has been piggybacked (478ms)
      ✓ should fail when there is no block for the exit position to enqueue (551ms)
      ✓ should fail when not called by the exit target of the output (419ms)
      When piggyback successfully
        ✓ should enqueue with correct data when it is the first piggyback of the exit on the token
        ✓ should NOT enqueue with correct data when it is not the first piggyback of the exit on the token (212ms)
        ✓ should set the exit as piggybacked on the output index (94ms)
        ✓ should set a proper piggyback bond size (106ms)
        ✓ should set the correct exit target to withdraw data on the output of exit data
        ✓ should emit InFlightExitOutputPiggybacked event

  Contract: PaymentProcessInFlightExit
    processInFlightExit
      ✓ should omit the exit if the exit does not exist (49ms)
      When bond return call failed
        on start ife bond return
          ✓ should not return bond
          ✓ should publish an event that bond return failed
        on piggyback bond return
          when transaction is non-canonical
            ✓ should not return piggyback bond
            ✓ should publish an event that input bond return failed
          when transaction is canonical
            ✓ should not return piggyback bond
            ✓ should publish an event that input bond return failed
      When any in-flight exit is processed successfully
        ✓ should transfer exit bond to the IFE bond owner if all piggybacked inputs/outputs are cleaned up (303ms)
        ✓ should only clean the piggyback flag of the inputs/outputs with same token (108ms)
        When not all piggybacks are resolved
          ✓ should NOT transfer exit bond to the IFE bond owner
          ✓ should NOT delete the exit from storage (89ms)
        When all piggybacks are resolved
          ✓ should transfer exit bond to the IFE bond owner
          ✓ should delete the exit from storage (87ms)
      When any input is spent, given the challenge game result is canonical
        ✓ should be treated as non canonical (397ms)
      When the exit is non canonical, and some inputs/outputs are piggybacked
        ✓ should withdraw ETH from vault for the piggybacked input (305ms)
        ✓ should NOT withdraw fund from vault for the piggybacked but already spent input (330ms)
        ✓ should NOT withdraw fund from vault for the non piggybacked input (275ms)
        ✓ should withdraw ERC20 from vault for the piggybacked input (279ms)
        ✓ should return piggyback bond to the input owner (281ms)
        ✓ should only flag piggybacked inputs with the same token as spent (316ms)
        ✓ should NOT flag output as spent (340ms)
        ✓ should emit InFlightExitInputWithdrawn event (283ms)
      When the exit is canonical, and some inputs/outputs are piggybacked
        ✓ should withdraw ETH from vault for the piggybacked output (420ms)
        ✓ should NOT withdraw from fund vault for the piggybacked but already spent output (447ms)
        ✓ should NOT withdraw from fund vault for the non piggybacked output (395ms)
        ✓ should withdraw ERC20 from vault for the piggybacked output (352ms)
        ✓ should return piggyback bond to the output owner (370ms)
        ✓ should flag ALL inputs with the same token as spent (451ms)
        ✓ should only flag piggybacked output with the same token as spent (411ms)
        ✓ should emit InFlightExitOutputWithdrawn event (342ms)

  Contract: PaymentStandardExitRouter
    processStandardExit
      ✓ should not process the exit when such exit is not exitable (120ms)
      ✓ should not process the exit when output already flagged as spent (155ms)
      ✓ should flag the output spent when sucessfully processed (128ms)
      ✓ should return standard exit bond to exit target when the exit token is ETH (118ms)
      ✓ should return standard exit bond to exit target when the exit token is ERC20 (149ms)
      ✓ should call the ETH vault with exit amount when the exit token is ETH (118ms)
      ✓ should call the Erc20 vault with exit amount when the exit token is an ERC 20 token (147ms)
      ✓ should deletes the standard exit data (140ms)
      ✓ should emit ExitFinalized event (121ms)
      when paying out bond fails
        ✓ should not pay out bond
        ✓ should publish an event informing that bond pay out failed

  Contract: PaymentStartInFlightExit
    startInFlightExit
      when calling start in-flight exit succeed with valid arguments
        ✓ should call the StateTransitionVerifier with correct arguments (1520ms)
        ✓ should store in-flight exit data (1581ms)
        ✓ should emit InFlightExitStarted event (1386ms)
        ✓ should charge user with a bond (1352ms)
        ✓ should succeed when first input transaction is the youngest (1447ms)
        ✓ should be able to start by non input or output owner (1394ms)
      when in-flight exit start is called but failed
        ✓ should fail when spending condition not registered (1736ms)
        ✓ should fail when spending condition not satisfied (1869ms)
        ✓ should fail when in-flight transaction is an invalid state transition (2343ms)
        ✓ should fail when state transition verification reverts (2337ms)
        ✓ should fail when any of input transactions is not of MoreVP protocol (1127ms)
        ✓ should fail when any of input transactions is not standard finalized (1472ms)
        ✓ should fail when the same in-flight exit is already started (2479ms)
        ✓ should fail when the in-flight tx is not the supported tx type (1009ms)
        ✓ should fail when there too many input transaction utxos provided (994ms)
        ✓ should fail when number of input utxos does not match the number of in-flight tx inputs (988ms)
        ✓ should fail when not called with a valid exit bond (116ms)
        ✓ should fail when there are no input transactions provided (897ms)
        ✓ should fail when number of input transactions does not match number of input utxos positions (767ms)
        ✓ should fail when number of input transactions does not match in-flight transactions number of inputs (1113ms)
        ✓ should fail when number of witnesses does not match in-flight transactions number of inputs (1027ms)
        ✓ should fail when number of merkle inclusion proofs does not match in-flight transactions number of inputs (990ms)
        ✓ should fail when in-flight tx input transactions are not unique (1066ms)

  Contract: PaymentStartStandardExit
    startStandardExit
      ✓ should fail when the transaction is not standard finalized (914ms)
      ✓ should fail when exit with amount of 0 (520ms)
      ✓ should fail when the exiting tx type is not the supported one (629ms)
      ✓ should fail when the block of the position does not exists in the Plasma Framework (543ms)
      ✓ should fail when amount of bond is invalid (90ms)
      ✓ should fail when not initiated by the output owner (630ms)
      ✓ should fail when same Exit has already started (1478ms)
      ✓ should fail when exit has already been spent (978ms)
      ✓ should charge the bond from the user (573ms)
      ✓ should save the correct StandardExit data when successfully done for deposit tx (614ms)
      ✓ should save the correct StandardExit data when successfully done for non deposit tx (607ms)
      ✓ should put the exit data into the queue of framework (600ms)
      ✓ should emit ExitStarted event (569ms)

  Contract: PaymentOutputToPaymentTxCondition
    verify
      ✓ should fail when input tx does not match the supported type of payment tx (217ms)
      ✓ should fail when spending tx does not match the supported type of payment tx (387ms)
      ✓ should fail when spending tx does not point to the utxo pos in input (316ms)
      ✓ should fail when failed to recover the signer from the signature (392ms)
      ✓ should fail when spending tx not correctly signed by the input owner (418ms)
      ✓ should return true when all verification passes (418ms)

  Contract: SpendingConditionRegistry
    spendingConditions
      ✓ should get empty address if not registered
    registerSpendingCondition
      ✓ should be able to register successfully (45ms)
      ✓ should reject when not registered by owner
      ✓ should not be able to register after renouncing the ownership (53ms)
      ✓ should reject when trying to register with output type 0
      ✓ should reject when trying to register with spending tx type 0 (74ms)
      ✓ should reject when trying to register with a non-contract address
      ✓ should reject when the (output type, spending tx type) pair is already registered (89ms)

  Contract: BondSize
    with normal cases
      ✓ should return the initial bond size
      ✓ should be able to update the bond to upperBoundMultiplier times its current value
      ✓ should fail to update bond to more than upperBoundMultiplier times its current value (38ms)
      ✓ should be able to update the bond to lowerBoundDivisor of its current value
      ✓ should fail to update bond to less than lowerBoundDivisor of its current value
      ✓ should not update the actual bond value until after the waiting period (56ms)
      ✓ should update the actual bond value after the waiting period (58ms)
      ✓ should update the actual bond value after the waiting period (117ms)
      ✓ should be able to update continuosly (137ms)
    with boundary size of numbers
      ✓ should able to update to max number of uint128 without having overflow issue (103ms)
      ✓ should be able to update to 1 (91ms)
      ✓ should NOT be able to update to 0 (64ms)

  Contract: ExitId
    isStandardExit
      ✓ should return true given a standard exit id for deposit tx (40ms)
      ✓ should return true given a standard exit id for non deposit tx
      ✓ should return false given an in-flight exit id
    getStandardExitId
      ✓ should get the correct exit id for deposit tx output
      ✓ should return distinct exit ids for deposits that differ only in utxo pos (57ms)
      ✓ should get the correct exit id for non deposit tx output
      ✓ should overflow when created a tx with more than 255 outputs (61ms)
    getInFlightExitId
      ✓ should get correct in-flight exit id

  Contract: ExitableTimestamp
    calculate
      ✓ should get the correct exit timestamp for deposit tx (38ms)
      ✓ should get the correct exit timestamp for non deposit tx whose age is older than MIN_EXIT_PERIOD
      ✓ should get the correct exit timestamp for non deposit tx whose age is younger than MIN_EXIT_PERIOD

  Contract: MoreVpFinalization
    isStandardFinalized
      ✓ should revert for invalid transaction position (91ms)
      ✓ should revert when the tx type is not registered to the framework yet (144ms)
      Given MVP protocol
        ✓ should revert as it is not supported by this library (190ms)
      Given MoreVP protocol
        ✓ should revert if there is no block root hash data in the PlasmaFramework for the position (172ms)
        ✓ should return true given valid inclusion proof (232ms)
        ✓ should return false given empty inclusion proof (211ms)
        ✓ should return false given invalid inclusion proof (208ms)
    isProtocolFinalized
      ✓ should revert when the tx type is not registered to the framework yet (158ms)
      Given MVP protocol
        ✓ should revert as it is not supported by this library (177ms)
      Given MoreVP protocol
        ✓ should return true when tx exist (148ms)
        ✓ should return true when tx is empty (80ms)

  Contract: OutputId
    computeDepositOutputId
      ✓ should get the correct output id for deposit tx output
      ✓ should return distinct output ids for deposits that differ in utxo pos (38ms)
    computeNormalOutputId
      ✓ should get the correct output id for non deposit tx output when output index is 0
      ✓ should get the correct output id for non deposit tx output when output index is not 0

  Contract: BlockController
    constructor
      ✓ should set the authority correctly (85ms)
      ✓ nextChildBlock is set to "childBlockInterval" (59ms)
      ✓ nextDeposit set to 1 (58ms)
      ✓ childBlockInterval is set as the inserted value (55ms)
    activateChildChain
      before activate
        ✓ should not be able to submit child chain block (92ms)
        ✓ should not be able to submit deposit block (101ms)
        ✓ should not be able to be activated by non authority (84ms)
        after activated by authority
          ✓ should change isChildChainActivated flag to true
          ✓ should emit ChildChainActivated event
          ✓ should not be able to activate again
    submitBlock
      ✓ saves the child chain block root to contract (103ms)
      ✓ updates "nextChildBlock" with a jump of "childBlockInterval" (114ms)
      ✓ resets "nextDeposit" back to 1 (129ms)
      ✓ emits "BlockSubmitted" event (84ms)
      ✓ reverts when not called by authority (84ms)
    submitDepositBlock
      ✓ saves the deposit block root to contract (148ms)
      ✓ adds 1 to nextDeposit (159ms)
      ✓ does not change nextChildBlock (108ms)
      ✓ reverts when exceed max deposit amount (childBlockInterval) between two child chain blocks (227ms)
      ✓ should revert when called from a newly registered (still quarantined) vault (179ms)
      ✓ reverts when not called by registered vault (81ms)
    isDeposit
      ✓ should return false when it is not a deposit block (93ms)
      ✓ should return true when it is a deposit block (93ms)
      ✓ should revert if the block does not exist (61ms)

  Contract: ExitGameController
    addExitQueue
      ✓ generates a priority queue instance to the exitsQueues map
      ✓ emits event
      ✓ reverts when adding the same queue (41ms)
      ✓ reverts when adding queue with vault id 0
    hasExitQueue
      ✓ returns true when queue is added (62ms)
      ✓ returns false when queue not added
    enqueue
      ✓ rejects when not called from exit game contract (77ms)
      ✓ rejects when such token has not been added yet (143ms)
      ✓ rejects when called from a newly registered (still quarantined) exit game (204ms)
      ✓ can enqueue with the same exitable timestamp (priority) multiple times (232ms)
      when successfully enqueued
        ✓ inserts the new priority to the queue (53ms)
        ✓ saves the exit data to map
        ✓ emits an ExitEnqueued event
        ✓ should be able to find the exit in the priority queue given exitId (48ms)
        ✓ should provide convenience method to show top exit priority
    processExits
      ✓ rejects when such token has not been added yet (109ms)
      ✓ rejects when the exit queue is empty (90ms)
      ✓ rejects when the top exit id mismatches with the specified one (194ms)
      ✓ does not process exit that is not able to exit yet (165ms)
      given the queue already has an exitable exit
        ✓ should be able to process when the exitId is set to 0 (75ms)
        ✓ should be able to process when the exitId is set to the exact top of the queue (88ms)
        ✓ should call the "processExit" function of the exit processor when processes (83ms)
        ✓ should delete the exit data after processed (113ms)
        ✓ should stop to process when queue becomes empty (81ms)
      given multiple exits with different priorities in queue
        ✓ should process with the order of priority and delete the processed exit from queue (132ms)
        ✓ should process no more than the "maxExitsToProcess" limit (82ms)
    isAnyOutputsSpent
      ✓ should return true when checking a spent output (53ms)
      ✓ should return false when checking an unspent output
      ✓ should return true when all of the outputs are spent (58ms)
      ✓ should return true when one of the outputs is spent (56ms)
      ✓ should return false when all of the outputs are not spent
   batchFlagOutputsSpent
      ✓ should be able to flag a single output (58ms)
      ✓ should be able to flag multiple outputs (78ms)
      ✓ should fail when try to flag with empty outputId (63ms)
      ✓ should fail when not called by Exit Game contracts
      ✓ should revert when called on quarantined Exit Game contract (156ms)
      ✓ should fail when reentrancy attack on processExits happens (332ms)
   flagOutputSpent
      ✓ should be able to flag an output (48ms)
      ✓ should fail when try to flag withempty outputId (43ms)
      ✓ should fail when not called by Exit Game contracts
      ✓ should revert when called on quarantined Exit Game contract (140ms)

 Contract: PlasmaFramework
   constructor
      ✓ should set the min exit period
      ✓ should set maintainer address

 Contract: Protocol
   MORE_VP()
      ✓ should return protocol value of MoreVP
   MVP()
      ✓ should return protocol value of MVP
   isValidProtocol
      ✓ should return true for MVP protocol
      ✓ should return true for MoreVP protocol
      ✓ should return false for invalid protocol

 Contract: ExitGameRegistry
   onlyFromNonQuarantinedExitGame
      ✓ should revert when the exit game contract is still quarantined
      ✓ accepts call when called by registered exit game contract on passed quarantine period
      ✓ reverts when not called by registered exit game contract
   registerExitGame
      ✓ should be able to register for both MVP and MoreVP protocol (143ms)
      ✓ rejects when not registered by maintainer
      ✓ rejects when trying to register with tx type 0
      ✓ rejects when trying to register with a non contract address
      ✓ rejects when protocol value is not MVP or MoreVP value (167ms)
      ✓ rejects when the tx type is already registered (100ms)
      ✓ rejects when the the exit game address is already registered (73ms)
     When succeed
        ✓ should be able to receive exit game address with tx type
        ✓ should be able to receive tx type with exit game contract address
        ✓ should be able to receive protocol with tx type
        ✓ should emit ExitGameRegistered event

 Contract: VaultRegistry
   onlyFromNonQuarantinedVault
      ✓ should accept call when called by registered and non quarantined vault contract
      ✓ should revert when not called by registered vault contract
      ✓ should revert when the vault contract is still quarantined
   vaults
      ✓ can receive vault contract address with vault id
   vaultToId
      ✓ can receive vault id with vault contract address
   registerVault
      ✓ should save the vault data correctly (66ms)
      ✓ should emit VaultRegistered event
      ✓ rejects when not registered by maintainer
      ✓ rejects when trying to register with vault id 0
      ✓ rejects when trying to register with a non-contract address
      ✓ rejects when The vault ID is already registered (94ms)
      ✓ rejects when the the vault contract address is already registered (68ms)

 Contract: ExitPriority
   computePriority
     Given different exitableAt
        ✓ should be positive correlative with "exitable" no matter txPos and exit id combination (462ms)
     Given exitableAt is the same
        ✓ should be positive correlative with "txPos" no matter how exit ids are (127ms)
     Given both exitableAt and txPos are the same
        ✓ should be positively correlative with exid id
   parseExitableAt
      ✓ should be able to parse the "exitableAt" from priority given exit id is 0
      ✓ should be able to parse the "exitableAt" from priority given max exit id value
      ✓ should be able to parse the "exitableAt" from priority given exitable timestamp is 0
      ✓ should be able to parse the "exitableAt" from priority given max exitable timestamp of uint42
      ✓ should be able to parse the "exitableAt" from priority given txPos is 0
      ✓ should be able to parse the "exitableAt" from priority given max txPos of uint54
      ✓ should be able to parse exit id from priority

 Contract: PriorityQueue
   getMin
      ✓ can get the min value (70ms)
      ✓ fails when empty
   currentSize
      ✓ can get current size (48ms)
   insert
      ✓ can insert one value (56ms)
      ✓ can insert two values with order (84ms)
      ✓ can insert multiple values out of order and still get the minimal number (255ms)
      ✓ is not idempotent (86ms)
      ✓ should fail when not inserted by the framework (who deploys the priority queue contract)
   delMin
      ✓ can delete when single value in queue (79ms)
      ✓ can delete when two values in queue (126ms)
      ✓ can delete when multiple values in queue (317ms)
      ✓ can delete all (474ms)
      ✓ should fail when not deleted by the framework (who deploys the priority queue contract)
      ✓ should fail when queue is empty (78ms)

 Contract: FungibleTokenOutputModel
   decodeOutput
      ✓ should return a fungible token output (105ms)
      ✓ should fail when output has not enough items (64ms)
      ✓ should fail when output has too many items (70ms)
      ✓ should fail when amount is 0 (75ms)
      ✓ should fail when outputGuard is 0 (81ms)
   getOutput
      ✓ should return a transaction output (160ms)
      ✓ should fail when output index is out of bounds (124ms)
```

```
Contract: GenericTransaction
  decode
     ✓ should decode a correctly formatted transaction (137ms)
     ✓ should decode a correctly formatted transaction with no inputs or outputs (79ms)
     ✓ should decode a correctly formatted transaction different output types (173ms)
     ✓ should decode a correctly formatted transaction with extra txData (128ms)
     ✓ should fail when not enough items (51ms)
     ✓ should fail when too many items (65ms)
     ✓ should fail when tx type is 0 (61ms)
     ✓ should fail when output type is 0 (96ms)
     ✓ should fail when not enough items in output (88ms)
     ✓ should fail when too many items in output (102ms)
     ✓ should fail when txType is a list (62ms)
     ✓ should fail when inputs is not a list (101ms)
     ✓ should fail when ouputs is not a list of lists (82ms)
     ✓ should fail when metadata is not 32 bytes (99ms)
  getOutput
     ✓ should return a transaction output (119ms)
     ✓ should fail when output index is out of bounds (123ms)

Contract: PaymentTransactionModel
  decode
     ✓ should decode payment transaction (263ms)
     ✓ should decode payment transaction with 4 inputs and 4 outputs (473ms)
     ✓ should fail when decoding transaction have more inputs than MAX_INPUT limit (107ms)
     ✓ should fail when decoding transaction have more outputs than MAX_OUTPUT limit (280ms)
     ✓ should fail when transaction does not contain metadata (50ms)
     ✓ should fail when decoding invalid transaction (50ms)
     ✓ should fail when the transaction contains a null input (164ms)
     ✓ should fail when the transaction type is zero (62ms)
     ✓ should fail when an output type is zero (155ms)
     ✓ should fail when an output amount is zero (232ms)
     ✓ should fail when txData is not zero (168ms)
     ✓ should fail when the transaction has no outputs (91ms)
  getOutputOwner
     ✓ should parse the owner address from output guard when output guard holds the owner info directly
  getOutput
     ✓ should get output (235ms)
     ✓ should fail when output index is out of bounds (262ms)

Contract: PaymentEip712Lib
  hashTx
     ✓ should hash normal transaction correctly (228ms)
     ✓ should hash deposit transaction correctly (195ms)
     ✓ should hash transaction correctly given transaction has empty metaData (219ms)

Contract: Bits
  bit operations
     ✓ should set bit on specified position (132ms)
     ✓ should set bit - small numbers (50ms)
     ✓ setting already set bit does not change the value
     ✓ should clear bit on specified position (94ms)
     ✓ should clear bit - small numbers (51ms)
     ✓ clearing already cleared bit does not change the value
     ✓ should answer whether bit is set (168ms)

Contract: FailFastReentrancyGuard
   ✓ should not allow local recursion (51ms)
   ✓ should not allow remote reentrancy but should not fail the top call (53ms)

Contract: Merkle
   ✓ check membership for tree where some leaves are empty
  checkMembership
     ✓ should return false when not able to prove included (49ms)
     ✓ should return false when a node preimage is provided as a leaf (50ms)
     ✓ should return false when leaf index mismatches the proof (45ms)
     ✓ should reject call when proof data size is incorrect
    Prove inclusion
       ✓ should return true for index 0 (51ms)
       ✓ should return true for index 1 (46ms)
       ✓ should return true for index 2 (45ms)
       ✓ should return true for index 3 (48ms)
       ✓ should return true for index 65534 (52ms)
       ✓ should return true for index 65535 (49ms)

Contract: OnlyWithValue
  onlyWithValue
     ✓ should accept call when the value matches
     ✓ should reject call when value mismatches

Contract: PosLib
  toStrictTxPos
     ✓ should parse a correct tx position
  getTxPostionForExitPriority
     ✓ should get a correct tx position for exit priority
  encode
     ✓ should correctly encode a position
     ✓ should fail when output index is too big
     ✓ should fail when transaction index is too big
     ✓ should fail when block number is too big
  decode
     ✓ should decode a position

Contract: Quarantine
  contract is quarantined
     ✓ should return true from isQuarantined when the contract is quarantined
     ✓ should return false from isQuarantined when the quarantine period has passed
  quarantineContract errors
     ✓ should revert when attempting to quarantine an empty address
     ✓ should revert when attempting to quarantine a contract again
  initial immune contract is not quarantined
     ✓ should return true from the initial immune contract
     ✓ should revert from the non immune contract

Contract: RLP
   ✓ should decode bytes32 (45ms)
   ✓ should fail decode to a bytes32 that has less than 32 bytes
   ✓ should decode bytes20
   ✓ should not decode an invalid length address
   ✓ should not decode an invalid length address
   ✓ should not decode an invalid length bytes32
   ✓ should decode uint 0
   ✓ should decode uint 0
   ✓ should decode positive uint (41ms)
```

```
    ✓ should fail to decode 0x00 as a uint
    ✓ should decode positive int
    ✓ should decode array (43ms)
    ✓ should fail to decode a list as a uint
    ✓ should fail to decode a list as an address
    ✓ should fail to decode a list as an bytes32

  Contract: RLP invalid tests
    ✓ should revert on invalid test wrongEmptyString (56ms)
    ✓ should revert on invalid test wrongSizeList
    ✓ should revert on invalid test wrongSizeList2
    ✓ should revert on invalid test incorrectLengthInArray
    ✓ should revert on invalid test randomRLP
    ✓ should revert on invalid test bytesShouldBeSingleByte00
    ✓ should revert on invalid test bytesShouldBeSingleByte01
    ✓ should revert on invalid test bytesShouldBeSingleByte7F
    ✓ should revert on invalid test leadingZerosInLongLengthArray1
    ✓ should revert on invalid test leadingZerosInLongLengthArray2
    ✓ should revert on invalid test leadingZerosInLongLengthList1
    ✓ should revert on invalid test leadingZerosInLongLengthList2
    ✓ should revert on invalid test nonOptimalLongLengthArray1
    ✓ should revert on invalid test nonOptimalLongLengthArray2
    ✓ should revert on invalid test nonOptimalLongLengthList1
    ✓ should revert on invalid test nonOptimalLongLengthList2
    ✓ should revert on invalid test shortStringInvalidEncodedLength1
    ✓ should revert on invalid test shortStringInvalidEncodedLength2
    ✓ should revert on invalid test stringListInvalidEncodedLength1
    ✓ should revert on invalid test stringListInvalidEncodedItemLength1
    ✓ should revert on invalid test stringListInvalidEncodedItemLength2 (38ms)
    ✓ should revert on invalid test invalidAddress
    ✓ should revert on invalid test invalidList
    ✓ should revert on invalid test emptyLonglist
    ✓ should revert on invalid test longstringInvalidEncodedLength1
    ✓ should revert on invalid test leadingZerosInUInt
    ✓ should revert on invalid test leadingZerosInUInt2
    ✓ should revert on invalid test leadingZerosInUInt3

  Contract: RLP official tests
    ✓ should pass emptystring
    ✓ should pass bytestring00
    ✓ should pass bytestring01
    ✓ should pass bytestring7F
    ✓ should pass shortstring
    ✓ should pass shortstring2
    ✓ should pass longstring
    ✓ should pass longstring2 (58ms)
    ✓ should pass zero
    ✓ should pass smallint
    ✓ should pass smallint2
    ✓ should pass smallint3
    ✓ should pass smallint4
    ✓ should pass mediumint1
    ✓ should pass mediumint2
    ✓ should pass mediumint3
    ✓ should pass mediumint4
    ✓ should pass mediumint5
    ✓ should pass emptylist
    ✓ should pass stringlist (116ms)
    ✓ should pass multilist (183ms)
    ✓ should pass shortListMax1 (388ms)
    ✓ should pass longList1 (580ms)
    ✓ should pass longList2 (4267ms)
    ✓ should pass listsoflists (176ms)
    ✓ should pass listsoflists2 (307ms)
    ✓ should pass dictTest1 (389ms)
    ✓ should pass bigint
    ✓ should pass bigint2

  Contract: SafeEthTransfer
    transferRevertOnError
      ✓ should revert when failed to transfer the fund
      ✓ should transfer the fund when successfully called
    transferReturnResult
      ✓ should revert when remaining gas is less than the gas stipend
      ✓ should return false when the contract does not have enough fund to transfer (40ms)
      ✓ should return false when called to a attacking contract that would fail in fallback function (94ms)
      ✓ should return false when called to a contract that would need more gas than gas stipend (90ms)
      when successfully called
        ✓ should return true when successfully transferred
        ✓ should transfer the fund

  Contract: Erc20Vault
    deposit
      before deposit verifier is set
        ✓ should fail with error message (54ms)
      after all contracts are set
        ✓ should store erc20 deposit (310ms)
        ✓ should emit deposit event (318ms)
        ✓ should spend erc20 tokens from depositing user (329ms)
        ✓ should be able to deposit when approved balance is more than deposit value (301ms)
        ✓ should not store a deposit when the tokens have not been approved (391ms)
        ✓ should not store a deposit when the tokens have not been approved enough value (384ms)
        ✓ should not store a deposit from user who does not match output address (356ms)
        ✓ should not store an ethereum deposit that sends funds
        ✓ should not store an ethereum deposit that does not send funds (343ms)
        ✓ should not accept a deposit with invalid output type (375ms)
        ✓ should not accept transaction that does not match expected transaction type (387ms)
        ✓ should not accept transaction with inputs (392ms)
        ✓ should not accept transaction with more than one output (505ms)
    deposit from NonCompliantERC20
      ✓ should store erc20 deposit (304ms)
    withdraw
      ✓ should fail when not called by a registered exit game contract (42ms)
      ✓ should transfer ERC token to the receiver (98ms)
      ✓ should emit Erc20Withdrawn event correctly (60ms)
      given quarantined exit game
        ✓ should fail when called under quarantine (178ms)
        ✓ should succeed after quarantine period passes (72ms)
    withdraw with NonCompliantERC20
      ✓ should transfer ERC token to the receiver (84ms)

  Contract: EthVault
    deposit
      before deposit verifier has been set
```

```
        ✓ should fail with error message
      after all related contracts set
        ✓ should store ethereum deposit (272ms)
        ✓ should emit deposit event (262ms)
        ✓ should charge eth from depositing user (254ms)
        ✓ should not store deposit when output value mismatches sent wei (739ms)
        ✓ should not store a deposit from user who does not match output address (386ms)
        ✓ should not store a non-ethereum deposit (349ms)
        ✓ should not accept a deposit with invalid output type (349ms)
        ✓ should not accept transaction that does not match expected transaction type (357ms)
        ✓ should not accept transaction with inputs (405ms)
        ✓ should not accept transaction with more than one output (531ms)
    withdraw
      ✓ should fail when not called by a registered exit game contract (45ms)
      ✓ should transfer ETH to the receiver (48ms)
      ✓ should emit EthWithdrawn event correctly (45ms)
      when fund transfer fails
        ✓ should emit WithdrawFailed event
        ✓ should not transfer ETH
      given quarantined exit game
        ✓ should fail when called under quarantine (59ms)
        ✓ should succeed after quarantine period passes (58ms)

  Contract: Vault
    setDepositVerifier / getEffectiveDepositVerifier
      ✓ should not allow for setting empty address as deposit verifier (44ms)
      ✓ should fail when not set by maintainer (41ms)
      before any deposit verifier is set
        ✓ should get empty address if nothing is set
      after the first deposit verifier is set
        ✓ should immediately take effect
        ✓ should emit SetDepositVerifierCalled event
        when setting the second verifier
          ✓ should not take effect before 2 minExitPeriod has passed
          ✓ should take effect after 2 minExitPeriod has passed
          ✓ should emit SetDepositVerifierCalled event


  648 passing (8m)
  1 pending
```

Code Coverage

The code features very good coverage.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
exits/fee/	100	100	100	100	
FeeClaimOutputToPaymentTxCondition.sol	100	100	100	100	
FeeExitGame.sol	100	100	100	100	
exits/interfaces/	100	100	100	100	
ISpendingCondition.sol	100	100	100	100	
IStateTransitionVerifier.sol	100	100	100	100	
exits/payment/	100	100	100	100	
PaymentExitDataModel.sol	100	100	100	100	
PaymentExitGame.sol	100	100	100	100	
PaymentExitGameArgs.sol	100	100	100	100	
PaymentInFlightExitModelUtils.sol	100	100	100	100	
PaymentTransactionStateTransitionVerifier.sol	100	100	100	100	
exits/payment/controllers/	100	98.06	100	100	
PaymentChallengeIFEInputSpent.sol	100	92.86	100	100	
PaymentChallengeIFENotCanonical.sol	100	97.06	100	100	
PaymentChallengeIFEOutputSpent.sol	100	91.67	100	100	
PaymentChallengeStandardExit.sol	100	90	100	100	
PaymentDeleteInFlightExit.sol	100	100	100	100	
PaymentPiggybackInFlightExit.sol	100	100	100	100	
PaymentProcessInFlightExit.sol	100	100	100	100	
PaymentProcessStandardExit.sol	100	100	100	100	
PaymentStartInFlightExit.sol	100	100	100	100	
PaymentStartStandardExit.sol	100	100	100	100	
exits/payment/routers/	**100**	**100**	**100**	**100**	
PaymentInFlightExitRouter.sol	100	100	100	100	

PaymentInFlightExitRouterArgs.sol	100	100	100	100	
PaymentStandardExitRouter.sol	100	100	100	100	
PaymentStandardExitRouterArgs.sol	100	100	100	100	
exits/payment/spendingConditions/	100	100	100	100	
PaymentOutputToPaymentTxCondition.sol	100	100	100	100	
exits/registries/	100	100	100	100	
SpendingConditionRegistry.sol	100	100	100	100	
exits/utils/	100	100	100	100	
BondSize.sol	100	100	100	100	
ExitId.sol	100	100	100	100	
ExitableTimestamp.sol	100	100	100	100	
MoreVpFinalization.sol	100	100	100	100	
OutputId.sol	100	100	100	100	
framework/	100	94.74	100	100	
BlockController.sol	100	100	100	100	
ExitGameController.sol	100	92.86	100	100	
PlasmaFramework.sol	100	100	100	100	
Protocol.sol	100	100	100	100	
framework/interfaces/	100	100	100	100	
IExitProcessor.sol	100	100	100	100	
framework/models/	100	100	100	100	
BlockModel.sol	100	100	100	100	
framework/registries/	100	100	100	100	
ExitGameRegistry.sol	100	100	100	100	
VaultRegistry.sol	100	100	100	100	
framework/utils/	100	100	100	100	
ExitPriority.sol	100	100	100	100	
PriorityQueue.sol	100	100	100	100	
Quarantine.sol	100	100	100	100	
transactions/	100	100	100	100	
FungibleTokenOutputModel.sol	100	100	100	100	
GenericTransaction.sol	100	100	100	100	
PaymentTransactionModel.sol	100	100	100	100	
transactions/eip712Libs/	100	100	100	100	
PaymentEip712Lib.sol	100	100	100	100	
utils/	100	100	100	100	
Bits.sol	100	100	100	100	
FailFastReentrancyGuard.sol	100	100	100	100	
Merkle.sol	100	100	100	100	
OnlyFromAddress.sol	100	100	100	100	
OnlyWithValue.sol	100	100	100	100	
PosLib.sol	100	100	100	100	
RLPReader.sol	100	100	100	100	
SafeEthTransfer.sol	100	100	100	100	
vaults/	100	100	100	100	
Erc20Vault.sol	**100**	**100**	**100**	**100**	
EthVault.sol	100	100	100	100	

Vault.sol	100	100	100	100	
vaults/verifiers/	100	100	100	100	
Erc20DepositVerifier.sol	100	100	100	100	
EthDepositVerifier.sol	100	100	100	100	
IErc20DepositVerifier.sol	100	100	100	100	
IEthDepositVerifier.sol	100	100	100	100	
All files	**100**	**98.85**	**100**	**100**	

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

```
1c4e30fd3aa765cb0ee259a29dead71c1c99888dcc7157c25df3405802cf5b09 ./contracts/Migrations.sol

53eacae398407de4a71f0ecaeb10a38f6127e21571b5b508035a4b02a31f3ac4
./contracts/src/exits/interfaces/IStateTransitionVerifier.sol

3e503c814e315a4b5c63f4f572766f5e82825a8b38a3df1a410d64c2d85fb01e ./contracts/src/exits/interfaces/ISpendingCondition.sol

b0c8e78075cdb155323bdfb3a42710d8e5e5bd9248d3987272b50b4306058821 ./contracts/src/exits/utils/ExitId.sol

8641dbadd022ca740bfa5be944c034152a2e77f223b70ff1df5289e956df5f90 ./contracts/src/exits/utils/ExitableTimestamp.sol

e616b2cc749ef6a9ddde3821bca21332b21f3705e74903db3d6293152bbd9f46 ./contracts/src/exits/utils/OutputId.sol

e6ac7bb1ca546fa93b92f71a57099b30bc94aac569ad12e911575829a0405a99 ./contracts/src/exits/utils/MoreVpFinalization.sol

84c9cbdfccd9e23c9615d7d0f649a0f5e088913653096732af4b9a41a857529b ./contracts/src/exits/utils/BondSize.sol

48dd7614961de0e76572017a8b6d3bc9271016b3c2dad75ec6ab5780dd8ec3e2
./contracts/src/exits/payment/PaymentTransactionStateTransitionVerifier.sol

463fe5b9119c991c803780a1e323db93071b59e282b080e6936017f27404b925 ./contracts/src/exits/payment/PaymentExitGameArgs.sol

27eb4b529e5d99d6275a9bc2795f7236755d94690d52c8d8ca343370a83d4e51 ./contracts/src/exits/payment/PaymentExitGame.sol

7c0314902d714ab04a996324bea50252b61b36ae53a86798af5540fef4111c5d
./contracts/src/exits/payment/PaymentInFlightExitModelUtils.sol

38f33c0577aa523a77cd2e4aa121b07558f3e82e0d0265dfb7225da2b39bca55 ./contracts/src/exits/payment/PaymentExitDataModel.sol

253a603760da47e6fee7f7c826dc60e58a6a9541191e73c3d1a06238c8a08955
./contracts/src/exits/payment/spendingConditions/PaymentOutputToPaymentTxCondition.sol

6f46d443c6ef92b9877b15d9d155787eb66ffcf11ef11c20758d10f6d03caa7e
./contracts/src/exits/payment/controllers/PaymentPiggybackInFlightExit.sol

a8c4d34a51ec3c9661332eafe13bae68a839725172f2812d4797de07939ef459
./contracts/src/exits/payment/controllers/PaymentChallengeIFENotCanonical.sol

0a50b9c9e0fc01e395b9c287ab728f7cf3ad14ca65514a9015dccd6b3d0a01a0
./contracts/src/exits/payment/controllers/PaymentStartInFlightExit.sol

ed09dccf17151cd5f4f165250f1a896147e3eaaaadd74f18de40547a86c2d992
./contracts/src/exits/payment/controllers/PaymentProcessInFlightExit.sol

071c35a76eeaa2dea437abc2afd212c17b7bd41e791268999fb1dfc6db1ba81e
./contracts/src/exits/payment/controllers/PaymentChallengeStandardExit.sol

a8f1bd4534570df05bef52b2ebd81065aaa0035c193a31d6591b65062f5b36ed
./contracts/src/exits/payment/controllers/PaymentChallengeIFEOutputSpent.sol

8f736b2c08afe6525f0da3abf6ea7d86a72a40a35110c1e6160e23af81246c56
./contracts/src/exits/payment/controllers/PaymentChallengeIFEInputSpent.sol

e22c9343c3d2b25542f3da45a6e5c999aa82858316ac57fed76fb503aed48e9b
./contracts/src/exits/payment/controllers/PaymentDeleteInFlightExit.sol

af3082bdb2991716e42549c4bc5c25a9b7a83d1fc4b4c15dcf6ffecc6ab0ebb1
./contracts/src/exits/payment/controllers/PaymentProcessStandardExit.sol

f8c29d33fd343e0677b765d2f9a73af98d6e0db810d5a90f7649bdc552124499
./contracts/src/exits/payment/controllers/PaymentStartStandardExit.sol

8ae92788506674858b7acec726a28a5c61474d203ca6d42f3afa439e013f7e09
./contracts/src/exits/payment/routers/PaymentStandardExitRouter.sol

9dcce8a08edad539cffb178217b0218c20b7cfc343f17599e8f585be46317ddf
./contracts/src/exits/payment/routers/PaymentInFlightExitRouter.sol

febf6e328ac2cba485af86d2a0e2424a8e3e4162216b0b7c18056b488efb50b5
./contracts/src/exits/payment/routers/PaymentStandardExitRouterArgs.sol

13b48f7010815cf5a243d50a1ca6b109ce94344288f6518514f958692d9695fc
./contracts/src/exits/payment/routers/PaymentInFlightExitRouterArgs.sol

8997cd1ba34111a37f52b75b5ef85c91a84d2dfaab2a866eda2b9b79ed960b9b
./contracts/src/exits/fee/FeeClaimOutputToPaymentTxCondition.sol

6b5713535b7d9c0ba2563718bec9aa4448d558991b5bc65ba1ebabbc9d163bb6 ./contracts/src/exits/fee/FeeExitGame.sol
```

```
da80c007fd44c140b404d47a62dcbd8105fd3dabbbcaee342d1758fe3913dd37
./contracts/src/exits/registries/SpendingConditionRegistry.sol

ee604a3d9800d1330ea6a32493bf1b2ddbd92a32073c4dc49b55eff5e2297ab5 ./contracts/src/vaults/Erc20Vault.sol

8369645827d895d97698a5763d09e604facfeff0a0780e47edf3fd4259d61000 ./contracts/src/vaults/EthVault.sol

147f9359c0020546f8493c1c9906ad5c1cfa9b8c1d728fef0e9bb6896e33de10 ./contracts/src/vaults/Vault.sol

6efbb58f6fb0eee8ad8a6f41ff73495d2410b6ce867bab98c2f6c99aab929d45 ./contracts/src/vaults/verifiers/Erc20DepositVerifier.sol

26f3aad86be11b6094b7dae1bd4fe77be98557e2106f490446a876b55842e39d ./contracts/src/vaults/verifiers/IErc20DepositVerifier.sol

e28639232f729b68909aa9a5946d4ddb72aeb0ae9f3a194e42f6a58a8b17b304 ./contracts/src/vaults/verifiers/IEthDepositVerifier.sol

e17b109e665d5db9e6adee530d271433f0cbb7047ce655c30e282bc9ddd40bbf ./contracts/src/vaults/verifiers/EthDepositVerifier.sol

fbb243682be3863ff859904b7fb62e4a48a43cacc101c5d5ff2fcc85b20dc5ad ./contracts/src/transactions/PaymentTransactionModel.sol

0b7c07e3e9a75b02a07e211c991fadacbae4d1629ec99c72076c7bbb68402cff ./contracts/src/transactions/GenericTransaction.sol

e8ce53ebb470c250d89cb9d6253c0a683ac9895a82f687941fff2d24a7150c47 ./contracts/src/transactions/FungibleTokenOutputModel.sol

d99ef1272a55d93f2be87493cb8156222dc3fbeedb2db1b4c4cfe8ba71809588
./contracts/src/transactions/eip712Libs/PaymentEip712Lib.sol

eac8d175d9aa497d85c7e387c229e4dd2649a40afe93967bcd97968e81ba98a4 ./contracts/src/utils/PosLib.sol

631ee6395d730c721d3d0d80d4ec8cf9bfe08bbe0b009de09b8a0a945c7785f5 ./contracts/src/utils/FailFastReentrancyGuard.sol

f64f9e84e2af7e50110dee8cfcea6907add54fe6969ed897892ef2cfeec1bb5d ./contracts/src/utils/RLPReader.sol

433ccb7c1ebe9928aaad140b9491a016164ae9c2ff230e5af5ed68e84aa33e64 ./contracts/src/utils/Bits.sol

3bc982014b8de3a37c0c651c8b78bfd31fb3dfe52792b8d40c299c812e249263 ./contracts/src/utils/OnlyFromAddress.sol

7c02b17fa0d26a20cc2f85f511cf1c574ecf3bec1a0bbe2f1397a7068887069b ./contracts/src/utils/SafeEthTransfer.sol

686e690ba698747970bd128436338f101f7bd41583ce7a862a828e92b7b8fc92 ./contracts/src/utils/OnlyWithValue.sol

116cfeab45f827ae3e2deb23e2d7b2f4429aff97a0c7983cd70964733f6e4587 ./contracts/src/utils/Merkle.sol

93e522f3ec04ac953a654f4a1234711a69378d4cdee8c3a534b09037f697631d ./contracts/src/framework/Protocol.sol

d75b4f86ffd3a9772668361349ff3a02a6cab3d98046d1ca9e0e762893da7d02 ./contracts/src/framework/PlasmaFramework.sol

4ff7ccdce7318d385f2590d1997e33cef1f16fbfa71f575dea95dafd830ba996 ./contracts/src/framework/BlockController.sol

ca1714d4665a77191cf8a09decefb2c42395fafedd3a6ce4c8a0b9a00b31dd36 ./contracts/src/framework/ExitGameController.sol

940bcafd5ec13619275e4cab3a91fefa15930db829167b9ca0238153af3869f8 ./contracts/src/framework/interfaces/IExitProcessor.sol

b734a60c842aa2e994d27cb1a2c7c3164c9c6943354e67de60c1b02c5813b754 ./contracts/src/framework/models/BlockModel.sol

6a784286dc848dddf718013b4e7bd7dd53ba7fe4d10b6377c10057b5e4f2c15e ./contracts/src/framework/utils/Quarantine.sol

f8711b6eabc80fce326609c9c682cac87718dfae40b70362501626eb7558c759 ./contracts/src/framework/utils/ExitPriority.sol

e9a072c2a10b46250d88239782a5f454ef59dc7a86603f8b725ea0061ff91daf ./contracts/src/framework/utils/PriorityQueue.sol

46b29360a28e4f83d293e6fb6d34b65e1f46cc2f2b57ef058a796fcd86ec6262 ./contracts/src/framework/registries/ExitGameRegistry.sol

f7afd9a6fec44b988b288989a096ffc30475b558369409cb6da2169e5472b17c ./contracts/src/framework/registries/VaultRegistry.sol

0bdfae2f3fbee26350cc3d7a449a0ff7cef4208a7722703e87a6f2db512f05ed ./contracts/mocks/Imports.sol

83687b5a87c1dee9622c6c16a2b8a60a5911eaa3c885d0455ea376f283efe719
./contracts/mocks/python_tests_wrappers/PriorityQueueTest.sol

ac441a183656f8a6a8003fc0bb916644fa84bbba1c4870e19310f94403372729 ./contracts/mocks/python_tests_wrappers/RLPTest.sol

ca226d32c9b5ed2dc2029f1068794a798c5cb9974f3ccb719e1649641b934167 ./contracts/mocks/exits/StateTransitionVerifierMock.sol

cc866ec5703b1d53de6af77bc5ff7c52d43dec3cbcc5d74753d4020ccf19ab9b ./contracts/mocks/exits/SpyPlasmaFrameworkForExitGame.sol

81b87bc3a4a3f2d38a8d0c8a9d63fe997feca3129430c38280cea935df69ab61 ./contracts/mocks/exits/SpendingConditionMock.sol

e587f32dca1244b4ecf1aa7d54f6013d3ea7fa1435807b10a929c6daf92f0227
./contracts/mocks/exits/utils/MoreVpFinalizationWrapper.sol

5619a0dee6e0ee30dff8107d7ca9effb5cd17e1cfdc11e56b1061938c2911175 ./contracts/mocks/exits/utils/OutputIdWrapper.sol

a80da6c21575d3d710bf316f667680d77b236692fd0bae0315f34a3f6a4c66ee ./contracts/mocks/exits/utils/ExitIdWrapper.sol

e76360818cb76b0dbd0be64af630e68756fc268342000cab0ec6e38f29d1241f ./contracts/mocks/exits/utils/ExitableTimestampWrapper.sol

eb49ff973146f7cf5139bf27559f17cd3cb5d44d10cb11ef8fc63db4f905afc1
./contracts/mocks/exits/payment/PaymentInFlightExitModelUtilsMock.sol

2a401ba5b8ce678dcae0360229c288fe2d8abf015d48d0782237d485385b0cda
./contracts/mocks/exits/payment/routers/PaymentStandardExitRouterMock.sol

ad6215186c73cbbe098d004cdc9fe1d4adb5d91e51fa3224b4909de8487afe09
./contracts/mocks/exits/payment/routers/PaymentInFlightExitRouterMock.sol

f93c21c0cd3460cacc560ad7a67b1b746e816d038e9d3f1ed738c4b12806b24a
./contracts/mocks/exits/dummyVaults/SpyErc20VaultForExitGame.sol

8b40c8d26ad831e1ea6875908bd07ccb76ea78441bb42f000632ae75f4780176
./contracts/mocks/exits/dummyVaults/SpyEthVaultForExitGame.sol

6cd9e5f1ecf3e87ce2343ae9e8ad244467e48bab2f49a6458d1a6c7ffd5cb1de ./contracts/mocks/attackers/OutOfGasFallbackAttacker.sol

3c1b36b076d6fbf2f351943349f96c2b297e29ee0e0d25b465f726e85bad3e30
./contracts/mocks/attackers/FallbackFunctionFailAttacker.sol

6b394d946840334d6fc99364b8f620ffec2fcd0f82e080fb141196ee0e9c983e
./contracts/mocks/attackers/FailFastReentrancyGuardAttacker.sol

7b6a87024e3d30a9bea2e20b638a4aa59e2b192b5393f77d585663917c0e9652 ./contracts/mocks/vaults/NonCompliantERC20.sol
```

```
dbcdb99266fb64b2d4d72dd43bcfe9133437055d91ac7799e00b071ad9aa555c
./contracts/mocks/transactions/GenericTransactionWrapper.sol

9364bb8afaff49a86d936d09c3a5a5ef749417c1358acc9448937249de01b45f
./contracts/mocks/transactions/PaymentTransactionModelMock.sol

e31618c8493a6932de5ee71a51e893abb4b271f6a03daf9e9af844889cfe3ac2
./contracts/mocks/transactions/FungibleTokenOutputWrapper.sol

85e8efe07b325460ac938d176e44d3e0d83d2371675ce1386f4f91132f43eace
./contracts/mocks/transactions/eip712Libs/PaymentEip712LibMock.sol

04b15354a37ef6c01ade81e061ed36a8f1e78edaef7eaedd6f816a7fe3f04900 ./contracts/mocks/utils/SafeEthTransferMock.sol

c862bc15d8708d5ff762826e061e6cde8f8607b75613294aed0a080cbc783c31 ./contracts/mocks/utils/BitsWrapper.sol

0c7691ccd05991709a55a9f328ec1dc7a8c76f1ee10549b638697f9dd95c089d ./contracts/mocks/utils/BondSizeMock.sol

585eb96d9a63b0886dc7bc7c7ba02f497f5a55eab5dff776d02173899e960779 ./contracts/mocks/utils/RLPMockSecurity.sol

11a7469cfd1d1bf436a72486b1f51e7f56af897a6553984a5cd61e119272f7a7 ./contracts/mocks/utils/MerkleWrapper.sol

c6f657fdc07ed42d4c7d2d368dc526b9ca35c88e9e154b66532876642740ad8d ./contracts/mocks/utils/RLPMock.sol

3b234f06a90b429635de492c2844e0be318feaa399505ca231fb75bd98973201 ./contracts/mocks/utils/PosLibWrapper.sol

fd18bc5c1d73bf9a03bad2c562eaa3842cd052646f7947b52d0ca835d5ad4627 ./contracts/mocks/utils/QuarantineMock.sol

6dc83b14b1457f885e89278f7fb846fdb28a15c7e1cacbfeecb4d20440ac4e2f ./contracts/mocks/utils/OnlyWithValueMock.sol

74cae5e5778a2ebf884f9a1b07592ca4b7d3c14a7629583b50ef1be31272061f ./contracts/mocks/framework/DummyExitGame.sol

29ec22fcb67e9587a6efb5cb5a72b78b9aa36b2ae6f8394eee918207bc2cf057 ./contracts/mocks/framework/DummyVault.sol

16ed08ae0b50a13b2285a157fb7b13552de693de6a53b93296ac3c481a4bfc34 ./contracts/mocks/framework/ExitGameControllerMock.sol

1d08082f4c18c04e75f514950f10fac1eade1ce4d3307d91b17f2afeb0dce058 ./contracts/mocks/framework/ProtocolWrapper.sol

2a1433de088a8202b054e20e6eca2b2f4baa39408f60933b5dfdaf46b38d5850 ./contracts/mocks/framework/ReentrancyExitGame.sol

b16384f32b5535b1340bc13b23998ba32b47b3049dc1501143292e3e765c3e41 ./contracts/mocks/framework/BlockControllerMock.sol

650a564724dc7685d0968297684c5510ecdabe0d3fdfd1d17ffcdc1d110ae047
./contracts/mocks/framework/utils/PriorityQueueLoadTest.sol

d9e73cc4a2baa746842c86a5ed9b38c13c3501731bbe08fa83f455ce2e3046c6 ./contracts/mocks/framework/utils/ExitPriorityWrapper.sol

3793e4cb5deaf0eaeec8a79902754fed439785d240b37ea4e6be9c172a4820bb
./contracts/mocks/framework/registries/VaultRegistryMock.sol

8c125fa5b49300ad9170c7ede221337cc75513dd867062a80c60e29467d619fc
./contracts/mocks/framework/registries/ExitGameRegistryMock.sol
```

Tests

```
68b68d22062528b2d01c1262ea7b74ee63d8501ec2ced4c4fe7e99a7ff8c3d2c ./test/src/exits/utils/OutputId.test.js

71a46d2da99691c0c50ba2fead6b0fd20338c095567e28afa6c8d1856fbe4690 ./test/src/exits/utils/ExitableTimestamp.test.js

950773d750a4a6d381e319e9b9f0848a07e0efb8905bb92de15d7046922b0ad2 ./test/src/exits/utils/BondSize.test.js

9ea7dc878d801edba25bd549a7d38783ed546842065f02e66b0dffc70c52c2b3 ./test/src/exits/utils/ExitId.test.js

c16594f661ee7d82c8c6a0b193f9a0f68583faed98a624e394e57466e57c7550 ./test/src/exits/utils/MoreVpFinalization.test.js

af7c6f2bdd580e419f4e314942bd4dd8fb0351cd4ca2c48413b0f36c3beab416
./test/src/exits/payment/PaymentExitGameReentrantProtected.test.js

d48b3cb69fd34f273c962a29cc7566a6b580a2f9a0e35500f3e69f6da4bb1bea ./test/src/exits/payment/PaymentInFlightExitRouter.test.js

91fd7750e597237bad76bed710bfaa324c7bd8e213c4f091163ebdc1618675af
./test/src/exits/payment/PaymentTransactionStateTransitionVerifier.test.js

f065c40268cdb9449b609cfa801b4e1d2aeb904e33666193c7844ba655dce42a ./test/src/exits/payment/PaymentExitGame.test.js

f03c81f986cfb4432a61a2bdba279897e16105978cb65cb76d51721cdb874fbf
./test/src/exits/payment/PaymentInFlightExitModelUtils.test.js

8c7f46174fac870893631c7b98512b25f99370877a6a08cd78d0617e5df19b2e ./test/src/exits/payment/PaymentUpdateBond.test.js

c10fdcce2a00108898fd9333bd83bf395710ddce51daa773b7807f03c5c5a4e4 ./test/src/exits/payment/PaymentStandardExitRouter.test.js

27566ccd1c114debd755661594049cf20f890d980764603689404e449d89655a
./test/src/exits/payment/spendingConditions/PaymentOutputToPaymentTxCondition.test.js

393979d279d790212846d8efa060c3abbfd296c553431e009e36f5868556bb9d
./test/src/exits/payment/controllers/PaymentPiggybackInFlightExitOnOutput.test.js

5cf2901ba74b4b76dde067af39377b3971bf8254f586d88fcecc13860aa10f0a
./test/src/exits/payment/controllers/PaymentChallengeIFENotCanonical.test.js

eb0e3be579ca72394deeb200438afe3b595452b279f5bc61478b9301efa2ad4c
./test/src/exits/payment/controllers/PaymentProcessInFlightExit.test.js

1b603741decfb377fda9d492d553805b9cccb96148a6692d5f17f266840f748c
./test/src/exits/payment/controllers/PaymentProcessStandardExit.test.js

a3226fbd8718df8f841404284bbdd473937ec24dee00c533c1af4fd90a9ae90b
./test/src/exits/payment/controllers/PaymentChallengeIFEOutputSpent.test.js

f22c240c2ac15b641bb11ad854901be5dc22a80ea0935ba40fe040d67c685e04
./test/src/exits/payment/controllers/PaymentStartInFlightExit.test.js

c775d598790e2c14d83d1400a366fed0055c705ecd0ee675b7a52179868ed9f1
./test/src/exits/payment/controllers/PaymentStartStandardExit.test.js

13822020808cc3b4a617ed9b7f93a2ce341aa7e4c61eff91d089e154f729f681
./test/src/exits/payment/controllers/PaymentDeleteInFlightExit.test.js
```

```
d8cbc1423ed3c9e997a12a54d9be0a3a4ccab77755ca700b7892cea201308786
./test/src/exits/payment/controllers/PaymentChallengeIFEInputSpent.test.js

197d7f8c37f9a2147fbfb00adeaed3508eef05a5a8252df84ab3969810a20bec
./test/src/exits/payment/controllers/PaymentChallengeStandardExit.test.js

87b544e22022b97ffb4669b7012d14c7604f4aad1921e53b4746fffd1225dd03
./test/src/exits/payment/controllers/PaymentPiggybackInFlightExitOnInput.test.js

bf6600e917b10f664f58dadd2161d4d12aeb1b416491303cf1c6483e7937236a
./test/src/exits/fee/FeeClaimOutputToPaymentTxCondition.test.js

6f5036653c00949a3104c4299e0f1dad55aecf3ccdf6b06ac08c27fd16d64272
./test/src/exits/registries/SpendingConditionRegistry.test.js

ad7f38c75edd71832e620d0d45b3eb67c57387e851fb2b31b2bf67fddd3722a2 ./test/src/vaults/Vault.test.js

b1d763b8e296bf4b1a61ac4c240aad0d9d7ffc4bf14e54e7ee17c99c5077de25 ./test/src/vaults/EthVault.test.js

dd9414ff1073c9560994cfdaf74e4d56ba0daa90b4d679d30f829528e853ec54 ./test/src/vaults/Erc20Vault.test.js

168da52a046dcbbb89c9bbbaa509f87620a40a3ab01db5ac1dcfd69ffe5a8f7e ./test/src/transactions/PaymentTransactionModel.test.js

14f457777106188e6b4f1604b4badffb4a7e819942308dc477d58d0f0e957a63 ./test/src/transactions/GenericTransaction.test.js

ffd3e88674d5c4e50e8bfed1573c74afb332f672f88614dae34927b83853ab50 ./test/src/transactions/FungibleTokenOutput.test.js

2f9e867cabd2c0129a977e7821d4199936bf052cb7ec72790c31b02048ba3325
./test/src/transactions/eip712Libs/PaymentEip712Lib.test.js

52ba4b83a3193f3a5d5dfbb7e382b32b4f9c43e24b085f8f789dc1f639b30f6c ./test/src/utils/RLP.test.js

395940e316811e94162d36c879bdead02b20250c15daed12caaf83dd47b1fb1c ./test/src/utils/Quarantine.test.js

26a11c8db36aec47f27fe40a70758379ad3bd657c1f37eed699972810c25b054 ./test/src/utils/RLPOfficial.test.js

48535a8acc827733af0863464d3822d888f3ef4d7403755a2aa7a76966755381 ./test/src/utils/Merkle.test.js

5bc2b07395c61eb0f159ff8d0e95428fdeb8a86d723a7e3578628a2e44e9de2a ./test/src/utils/OnlyWithValue.test.js

f5d0154228ea1d6f0cfc92a00635f4ddcfacb56f7f568eab980a0d99cebeb863 ./test/src/utils/PosLib.test.js

6dff5b1a1afe722145419bd114514204d0681b5a73f24734a1d7375dabb37486 ./test/src/utils/SafeEthTransfer.test.js

9d6845e99c36da84a982b924f3466473783aa3a094abfc5b014be7adfebf6920 ./test/src/utils/RLPInvalid.test.js

ef4e284e0b6452341eb76e60587ab6be8c51a5020682d4409d7562f64571b966 ./test/src/utils/BitsLib.test.js

07db22971cc5aa23f05fb282246e2c3252e75ec3ba3d9516e27fd0b630b5ea99 ./test/src/utils/FailFastReentrancyGuard.test.js

655dfee56d59d6d4246ec62a64968b2d6be13644f041ddd5f7445225d353a645 ./test/src/framework/ExitGameController.test.js

5b9c034028709c96c6ccf38c56613bf16dbe20829b83f4344f8f565c0656c217 ./test/src/framework/BlockController.test.js

7863d919bf6b2679c79ee8a93fdf21249f153054d0f9dcc8ed6873a061b0dd4e ./test/src/framework/PlasmaFramework.test.js

61e156e1711a8532b838ae1ca97b76675a0fa9d0593fc47b6a9b2ccfd9757e26 ./test/src/framework/Protocol.test.js

a93ab14332221bf20f21404fcfdd66652b3d05e074ad01c5992ec24582ec8d29 ./test/src/framework/utils/ExitPriority.test.js

da6b9d2d96a7fa83977b2b1883d5634f62b8f973c0c374eae2a45120c28fe44d ./test/src/framework/utils/PriorityQueue.test.js

8b8bda315334473f6587fd01d5600a02618258d54b3568956f408892a8b18f2b ./test/src/framework/registries/ExitGameRegistry.test.js

35722b865beebc79bd0edf0cca4cc90318449594b693541cbaed132f0e330cb6 ./test/src/framework/registries/VaultRegistry.test.js

ac9a8e56e20218fe126778c5a83e664c9e20c763bd1792ec2ce14a3ab31ddf05 ./test/helpers/constants.js

e7374804334e45bb65e5d23477eff24f35225bbe90fd7b4c084da71b29bd54d1 ./test/helpers/transaction.js

34063756bfac78272118a7fea834b6f3446252e9e25707f83f41f611c6b2c667 ./test/helpers/merkle.js

53ea3dc559bd24a60dfc8c2bfd05486330c04da402c8383ea0f3b085fa951116 ./test/helpers/positions.js

dc345cc076e34d385c00ef708b60901bda7467605d7a65405aba4754d4df4ea1 ./test/helpers/sign.js

ef82567a96836e2ff911c7ccf7de57efabfe1bdba4a9bc945dcea96e1b072872 ./test/helpers/paymentEip712.js

81b1b419df403a690d1304d2e01ce66b1b12bbede0e7be362d5b19ad03e6ac34 ./test/helpers/utils.js

79b2b62669ff69b15ec1810280ceff77be6691f2d20138e7372a3d5f26e1d209 ./test/helpers/exitable.js

45e8d0584893f5db9ee705c6c0ab39859043f7b085a6628c640e861be7a9449e ./test/helpers/ife.js

5e298ca383ed9c4cae83734e5882ef07ccb36ea298fa3b4b7d5b4aca4ae4089f ./test/helpers/testlang.js

dd161f672ee97fd399b61400612e6f8546c91823ca786eb09df70caba7f63cdf ./test/endToEndTests/PaymentInFlightExit.e2e.test.js

16a1cf60dfed9859bd7cab6c155ee29a5f0453f14b995d0b35effcb2ad1b18da ./test/endToEndTests/PaymentStandardExit.e2e.test.js

8784923c657763a7311fa6f7d0d4e8ba7c24945633be91c9a329f3fa9fcf1359 ./test/endToEndTests/StandardExit.load.test.js

f8aa6441c58806ef9e50d710359f7f6ab5d21077eb1891a5dfa717ed8f275076 ./test/endToEndTests/FeeClaim.e2e.test.js

7eb0e16135f6f551435318e6df886c682d888a2432971a636753de80f56171ab ./test/loadTests/PriorityQueue.load.test.js
```

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure smart contracts at scale using computer-aided reasoning tools, with a mission to help boost adoption of this exponentially growing technology.

Quantstamp's team boasts decades of combined experience in formal verification, static analysis, and software verification. Collectively, our individuals have over 500 Google scholar citations and numerous published papers. In its mission to proliferate development and adoption of blockchain applications, Quantstamp is also developing a new protocol for smart contract verification to help smart contract developers and projects worldwide to perform cost-effective smart contract security audits.

To date, Quantstamp has helped to secure hundreds of millions of dollars of transaction value in smart contracts and has assisted dozens of blockchain projects globally with its white glove security auditing services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Finally, Quantstamp's dedication to research and development in the form of collaborations with leading academic institutions such as National University of Singapore and MIT (Massachusetts Institute of Technology) reflects Quantstamp's commitment to enable world-class smart contract innovation.