Quantstamp Contract Security Certificate

October 18th 2019 — Quantstamp Verified

StablePay

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

Executive Summary

Type	Payment platform
Auditors	Leonardo Passos, Senior Research Engineer Alex Murashkin, Senior Software Engineer Sung-Shine Lee, Research Engineer
Timeline	2019-08-12 through 2019-10-11
EVM	Constantinople
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	

Repository	Commit
stablepay_contracts_dev	ff8e2

Total Issues	**12** (12 Resolved)
High Risk Issues	**2** (2 Resolved)
Medium Risk Issues	**3** (3 Resolved)
Low Risk Issues	0
Informational Risk Issues	0
Undetermined Risk Issues	**7** (7 Resolved)

Severity Categories	
High	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	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	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.

Changelog

- 2019-08-24 - Initial report (commit d4d07)
- 2019-10-26 - Fixes and diff analysis (commit 1f960)
- 2019-10-11 - Fixes and diff analysis (commit fe414)
- 2019-10-15 - Fixes and final report (commit ff8e2)

Quantstamp Audit Breakdown

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

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

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

Setup

Tool Setup:

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. Flattened the source code using `truffle-flattener` to accommodate the auditing tools.
6. Installed the Mythril tool from Pypi: `pip3 install mythril`
7. Ran the Mythril tool on each contract: `myth -x path/to/contract`
8. Ran the Securify tool: `java -Xmx6048m -jar securify-0.1.jar -fs contract.sol`
9. Installed the Oyente tool from Docker: `docker pull luongnguyen/oyente`
10. Migrated files into Oyente (root directory): `docker run -v $(pwd):/tmp - it luongnguyen/oyente`
11. Ran the Oyente tool on each contract: `cd /oyente/oyente && python oyente.py /tmp/path/to/contract`
12. Cloned the MAIAN tool: `git clone --depth 1 https://github.com/MAIAN-tool/MAIAN.git maian`
13. Ran the MAIAN tool on each contract: `cd maian/tool/ && python3 maian.py -s path/to/contract contract.sol`
14. Installed the Slither tool: `pip install slither-analyzer`
15. Run Slither from the project directory `slither .`

Assessment

Findings

Denial-of-Service (DoS)

Severity: *High*

Status: Fixed

Contract(s) affected: `contracts/base/StablePayStorage.sol, L168-172 (commit d4d07)`

Related Issue(s): SWC-128

Description: A Denial-of-Service (DoS) attack is a situation which an attacker renders a smart contract unusable. The view function `getExpectedRates()` iterates over the mapping of registered providers. While there is no security risk for the smart contract (this view function is not called by any function in StablePay's set of contracts), it could potentially cause a Denial of Service attack on other components not part of this audit (e.g., StablePay's frontend). This is due to the fact that `registerSwappingProvider()` is public and does not have any access control. That means that anybody can register any number of providers. Attack: (1) Hackers decide to bring a dApp that relies on `getExpectedRates` down; (2) Each hacker registers a large amount of providers, and keep doing that for a while; (3) Any dApp making use of `getExpectedRates()` will now freeze.

Recommendation: Put an access control modifier (e.g., `onlyOwner`) as part of the function definition of `registerSwappingProvider()`.

Missing Input Validation

Severity: *High*

Status: Fixed

Contract(s) affected: `contracts/base/Upgrade.sol: L45 (commit d4d07)`

Description: While the implementation correctly requires `oldContractAddress` to be different than `0x0`, it does not put the same restriction on `_upgradedContractAddress`. Therefore, the owner may inadvertently update `_upgradedContractAddress` to `0x0` and make the contract unusable afterwards.

Recommendation: Check if the new contract address is not `0x0`. If the intention is to kill the contract, create a specific function for that.

Reentrancy Protection Flaw

Severity: *Medium*

Status: Fixed

Contract(s) affected: `contracts/base/StablePayBase.sol, L560 (commit d4d07)`

Related Issue(s): SWC-107

Description: One of the modifiers in `requireTransferWithTokens` is `nonReentrant()`. Instead of protecting `requireTransferWithTokens` callers against reentrancy (e.g., `transferWithTokens`), the existing code is only protecting `requireTransferWithTokens`. Thus, the intended protection against reentrancy is not in place.

Exploit Scenario:

Recommendation: Move the nonReentrant modifier to `transferWithTokens` (the only function calling `requireTransferWithTokens`). Note: look at related (best practice) issue related to `contracts/base/StableBasePay.sol`, L539, which we detail later in the report.

Reentrancy Protection Flaw

Severity: *Medium*

Status: Fixed

Contract(s) affected: `contracts/base/StablePayBase.sol, L682 (commit d4d07)`

Related Issue(s): SWC-107

Description: One of the modifiers in `requireTransferWithEthers` is `nonReentrant()`. Instead of protecting `requireTransferWithEthers` callers against reentrancy (e.g., `transferWithEthers`), the existing code is only protecting `requireTransferWithEthers`. Thus, the intended protection against reentrancy is not in place.

Recommendation: Move the `nonReentrant` modifier to `transferWithEthers` (the only function calling `requireTransferWithEthers`). Note: look at related (best practice) issue related to `contracts/base/StableBasePay.sol, L574`, which we detail later in the report.

Inconsistency between Doc and Implementation

Severity: *Medium*

Status: Fixed

Contract(s) affected: `contracts/base/StablePayBase.sol, L574 (commit d4d07)`

Description: `transferWithTokens` basically performs a swap with the given input provider. The FAQ, however, states that the platform selects the best provider in terms of price and gas cost (see FAQ in https://stablepay.io/faqs, “How does StablePay work?”). We don’t see such behavior anywhere in the Solidity code. It is plausible, however, that such selection occurs elsewhere (e.g., in the js-part of the DApp).

Recommendation: Explicitly state what the current implementation does in the FAQ, adjusting the text accordingly.

Complicated Logic with Little Documentation

Severity: *Undetermined*

Status: Fixed

Contract(s) affected: `contracts/base/StablePayBase.sol, L136-156 (commit d4d07)`

Description: Logic in `getFeeAmount` is far from clear; it needs to be better documented so that others can understand its actual behavior.

Recommendation: Improve the implemented logic in `getFeeAmount`, adding supporting documentation to the code.

Use of Experimental Feature in Production

Severity: *Undetermined*

Status: Fixed

Contract(s) affected: `contracts/base/action/CompoundMintPostAction.sol, L2 (commit d4d07)`, `contracts/base/action/PostActionBase.sol, L2 (commit d4d07)`, `contracts/base/action/TransferToPostAction.sol, L2 (commit d4d07)`, `contracts/base/StablePayBase.sol, L2 (commit d4d07)`, `contracts/base/StablePayStorage.sol, L2 (commit d4d07)`, `contracts/interface/IPostAction.sol, L2 (commit d4d07)`, `contracts/interface/IProviderRegistry.sol, L2 (commit d4d07)`, `contracts/interface/IStablePay.sol, L2 (commit d4d07)`, `contracts/mock/base/action/PostActionBaseMock.sol, L2 (commit d4d07)`, `contracts/mock/base/StablePayBaseMock.sol, L2 (commit d4d07)`, `contracts/mock/base/StablePayStorageMock.sol, L2 (commit d4d07)`, `contracts/mock/provider/CustomSwappingProviderMock.sol, L2 (commit d4d07)`, `contracts/mock/provider/SwappingProviderMock.sol, L2 (commit d4d07)`, `contracts/providers/AbstractSwappingProvider.sol, L2 (commit d4d07)`, `contracts/providers/ISwappingProvider.sol, L2 (commit d4d07)`, `contracts/providers/KyberSwappingProvider.sol, L2 (commit d4d07)`, `contracts/providers/UniswapSwappingProvider.sol, L2 (commit d4d07)`

Description: Many files rely on the experimental feature `ABIEncoderV2`, whose use is discouraged in production. Some bugs associated with it have already been reported (see https://blog.ethereum.org/2019/03/26/solidity-optimizer-and-abiencoderv2-bug/ and https://blog.ethereum.org/2019/06/25/solidity-storage-array-bugs/), but have been fixed on release 0.5.10 onwards.

Recommendation: Migrate your contracts to any version higher or equal to 0.5.10.

Unclear Gas Reduction

Severity: *Undetermined*

Status: Fixed

Contract(s) affected: `contracts/base/proxy/DelegateProxy.sol, L23 (commit d4d07)`

Description: The assembly block in `delegatedFwd` contains a reduction on gas limit: `sub(gas, fwdGasLimit)`. It is unclear why that is needed.

Recommendation: If the gas instruction is indeed needed, then clarify the reason with improved documentation; otherwise, if not needed, just remove it.

Unknown Intent

Severity: *Undetermined*

Status: Fixed

Contract(s) affected: `contracts/providers/KyberSwappingProvider.sol, L245, 308 (commit d4d07)`

Description: `minRate` is never used. However, upon closer look to the documentation of the trade function in Kyber (https://developer.kyber.network/docs/API_ABI-KyberNetworkProxy/#trade), we found it expects a `minConversionRate`, and the call is canceled if the actual rate is lower. We assume that StablePay is optimizing for the maximum conversion rate, which would explain why `maxRate` is used instead of `minRate`.

Recommendation: Clarify your intent with proper documentation in code. Additionally, consider having support and API clarifications directly with the Kyber team.

Fallback Gas Stipend

Severity: *Undetermined*

Status: Fixed

Contract(s) affected: `contracts/base/Base.sol (commit d4d07)`

Description: The fallback function will exceed the 2300 gas stipend. Thus, if contracts inheriting from `Base` are to receive eth back, these transfers will fail unless they are performed using call(...).

Recommendation: Make sure the fallback implementation only emits an event and nothing else.
Update: all fallbacks have been rewritten as a means to not go above the 2300 gas stipend. The only exception is in `ProxyBase`. The latter shall not be triggered by `send(...)` or `transfer(...)`. Instead, the fallback in `ProxyBase` shall be triggered when users invoke a non-existent function in `ProxyBase`; in turn, the fallback will forward the call to the underlying upgradable contract (which should contain the target function). When invoking the non-existent function, callees can pass larger gas limits; thus, the fallback in `ProxyBase` is not subject to the same gas stipend as send and transfer. Hence, we consider this issue fixed altogether.

Source Token May Not Be `ETH_TOKEN_ADDRESS`

Severity: *Undetermined*

Status: Fixed

Contract(s) affected: `contracts/providers/KyberSwappingProvider.sol (commit 1f960)`

Description: In `swapEther`, it is unclear what would happen if `_order.sourceToken` is not set to `ETH_TOKEN_ADDRESS`.

Recommendation: Check whether `sourceToken` is `ETH_TOKEN_ADDRESS`.

Post Action May Be `0x0`

Severity: *Undetermined*

Status: Fixed

Contract(s) affected: `contracts/base/PostActionRegistry.sol (commit 1f960)`

Description: Either (or both) default post action and given input action (`postAction`) could be `0x0`, which could cause a logic error in the execution flow of a transfer.

Recommendation: Check that neither post action is `0x0`.

Test Results

Test Suite Results

```
------------------------------|----------|----------|----------|----------|----------------|
File                          |  % Stmts | % Branch |  % Funcs |  % Lines |Uncovered Lines |
------------------------------|----------|----------|----------|----------|----------------|
 contracts/                   |      100 |      100 |      100 |      100 |                |
  StablePay.sol               |      100 |      100 |      100 |      100 |                |
 contracts/base/              |    60.38 |    43.89 |    64.08 |     60.3 |                |
  Base.sol                    |    81.82 |    83.33 |    76.92 |    81.48 |... 140,143,156 |
  CompoundSettings.sol        |      100 |      100 |      100 |      100 |                |
  PostActionRegistry.sol      |    52.63 |       40 |    58.33 |    59.09 |... 158,159,160 |
  Registration.sol            |    53.33 |      100 |       80 |    53.33 |... 85,90,94,99 |
  Role.sol                    |    83.33 |    64.29 |       80 |       84 |144,185,190,194 |
  Settings.sol                |    90.63 |       75 |    83.33 |    90.63 |    162,230,233 |
  StablePayBase.sol           |    25.62 |    22.73 |    33.33 |     24.6 |... 736,739,747 |
  StablePayStorage.sol        |    91.43 |    73.33 |    84.21 |       92 |... ,95,138,266 |
  Storage.sol                 |    77.78 |    83.33 |    60.87 |    65.38 |... 227,243,251 |
  Upgrade.sol                 |      100 |       75 |      100 |      100 |                |
  Vault.sol                   |    13.04 |     7.69 |    42.86 |    13.04 |... 143,148,150 |
 contracts/base/action/       |       10 |        0 |    46.15 |     12.2 |                |
  CompoundMintPostAction.sol  |     5.26 |        0 |       25 |     5.26 |... 136,143,157 |
  EtherTransferPostAction.sol |     6.67 |        0 |       25 |     6.67 |... 131,134,136 |
  PostActionBase.sol          |      100 |      100 |      100 |      100 |                |
  TransferToPostAction.sol    |        0 |        0 |       50 |        0 |    42,49,53,60 |
 contracts/base/proxy/        |    73.33 |    41.67 |    85.71 |    76.47 |                |
  DelegateProxy.sol           |      100 |      100 |      100 |      100 |                |
  ERCAbstractProxy.sol        |      100 |      100 |      100 |      100 |                |
  IsContract.sol              |      100 |      100 |      100 |      100 |                |
  ProxyBase.sol               |       60 |     12.5 |       80 |       60 |    71,72,73,78 |
```

```
 contracts/interface/           |      100 |      100 |      100 |      100 |                |
  ICompoundSettings.sol         |      100 |      100 |      100 |      100 |                |
  IOwnable.sol                  |      100 |      100 |      100 |      100 |                |
  IPostAction.sol               |      100 |      100 |      100 |      100 |                |
  IPostActionRegistry.sol       |      100 |      100 |      100 |      100 |                |
  IProviderRegistry.sol         |      100 |      100 |      100 |      100 |                |
  IRegistration.sol             |      100 |      100 |      100 |      100 |                |
  IRole.sol                     |      100 |      100 |      100 |      100 |                |
  ISettings.sol                 |      100 |      100 |      100 |      100 |                |
  IStablePay.sol                |      100 |      100 |      100 |      100 |                |
  IStorage.sol                  |      100 |      100 |      100 |      100 |                |
  ISwappingProvider.sol         |      100 |      100 |      100 |      100 |                |
  IUpgrade.sol                  |      100 |      100 |      100 |      100 |                |
  IVault.sol                    |      100 |      100 |      100 |      100 |                |
 contracts/providers/           |      9.4 |     4.55 |       25 |     9.27 |                |
  AbstractSwappingProvider.sol  |       25 |    11.11 |    22.22 |       24 |... 155,157,162 |
  KyberSwappingProvider.sol     |    10.14 |     6.25 |    27.27 |       10 |... 355,358,365 |
  UniswapSwappingProvider.sol   |     1.79 |        0 |       25 |     1.79 |... 232,234,237 |
 contracts/util/                |    90.32 |    88.89 |    92.31 |    91.18 |                |
  AddressLib.sol                |       80 |      100 |    88.89 |    72.73 |       17,25,28 |
  Bytes32ArrayLib.sol           |    95.24 |       80 |      100 |      100 |                |
  StablePayCommon.sol           |      100 |      100 |      100 |      100 |                |
--------------------------------|----------|----------|----------|----------|----------------|
All files                       |    46.55 |     32.5 |    61.39 |    47.43 |                |
--------------------------------|----------|----------|----------|----------|----------------|
```

Automated Analyses

Maian

No issues were detected.

Oyente

No issues were detected.

Mythril

No issues were detected.

Securify

Except for the use of an experimental feature (which we included in this report), no other vulnerability was detected.

Slither

Issues were detected; after manually checking each, this report does include those that proved to be right.

Adherence to Best Practices

Best Practices to Follow (commit d4d07)

- `contracts/Base.sol`: `onlyAdmin` modifier is used in mocking (tests), but not elsewhere. Seems to have no purpose overall. Remove it. **Fixed**
- `contracts/base/StablePayBase.sol`, L38: `isSender()` should not require the sender parameter; it's a room for errors. The function can access `msg.sender` directly in its body. **Fixed**
- `contracts/base/StablePayBase.sol`, L39: Rename 'to' to 'target'. **Fixed**
- `contracts/base/StablePayBase.sol`, L52-53, 69-70, 74, 91, 246, 263, 268: negate message in `require` statement. **Fixed**
- `contracts/base/StablePayBase.sol`, L73: `areOrderAmountsValidEther` allows zero on sourceAmount. **Fixed**
- `contracts/base/StablePayBase.sol`, L103: consider renaming `STABLE_PAY_STORAGE_NAME` to `STABLE_PAY_PROVIDER_REGISTRY_NAME` to better reflect intent. **Fixed**
- `contracts/base/StablePayBase.sol`, L179: remove the bool return in `allowanceHigherOrEquals`, as the function always returns `true` on success or reverts otherwise. Moreover, callers currently don't check the return value anyways. This seems to be a general pattern throughtout the code and should be refactored altogether. **Fixed**
- `contracts/base/StablePayBase.sol`, L276, typo: `Base on that` => `Based on that` **Fixed**
- `contracts/base/StablePayBase.sol`, L353: 'to' as a target address is misleading here. This also occurs in other parts of the code. Rename accordingly. **Fixed**
- `contracts/base/StablePayBase.sol`, L456: The comment "// Transfer tokens from StablePayBase to swapping provider." is misleading - it actually transfers from `msg.sender` (the owner of the tokens) to the provider, not from StablePayBase. We advise checking intent and fixing the comment accordingly. **Fixed**
- `contracts/base/StablePayBase.sol`, L527-534: the else statement else `{ emitExecutionTransferFailedEvent (...)}is not necessary; rather,` emitExecutionTransferFailedEvent(...)` suffices. **Fixed**
- `contracts/base/StablePayBase.sol`, L539: `requireTransferWithTokens` has an unnecessary parameter: `providerKeys`. Remove it from signature and the no-op statement where it appears. **Fixed**
- `contracts/base/StablePayBase.sol`, L579: change `require` to `revert`, as there is no condition. Make this consistent within the code base. **Fixed**
- `contracts/base/StablePayBase.sol`: naming of `calculateAndTransferToAmount()` is misleading. Since the function actually transfers the tokens to the `postActionAddress`, we advise renaming it to `calculateAndTransferAmountToPostActionAddress`. **Fixed**
- `contracts/base/StablePayBase.sol`, L612: "// Transfer back the Ether left to the 'to' address." is misleading - it actually sends the Ether

• back to whoever sent the transaction (`msg.sender`). We recommend fixing the comment accordingly. **Fixed**

• `contracts/base/StablePayBase.sol`: `transferWithTokens` and `transferWithEthers` are public functions, but no documentation is provided for users (e.g., what do parameters mean). **Fixed**

• There are TODOs in multiple files. Please consider addressing them. **Fixed**

• `contracts/util/AddressLib.sol`: not used anywhere in the solidity codebase. Either use it or remove it entirely. **Fixed**

• Many files seem to have been copied from existing projects (`WETH9.sol`, `SafeMath.sol`). Consider using a dependency manager instead; if not, properly document the version of cloned files, linking them to the commit hash creating these files in the repositories they originally came from. This facilitates traceability and porting of bug fixes. **Fixed**

• Inside the interfaces folder, there are some contracts named after interfaces (e.g., `IERCProxy`) that are not interfaces in the strict sense of the word. According to Solidity's documentation (https://solidity.readthedocs.io/en/v0.5.3/contracts.html?highlight=interface#interfaces), an interface **must***: (1) not inherit other contracts or interfaces; (2) declare only external functions; (3) not declare a constructor; (4) not declare state variables. Rename any contract in the interface folder that does not satisfy any of the four properties mentioned as an abstract contract (e.g., `IERCProxy` => `ERCAbstractProxy`). Any other contract named after an interface and that satisfies all four properties should be declared explicitly with the interface keyword if not already done so. **Fixed**

• `contracts/providers/UniswapSwappingProvider.sol`, L75: initially setting allowance to zero actually doesn't have any impact and isn't necessary, because another approval is set as the next statement, and it's part of the same transaction **Fixed**

• `contracts/base/StablePayStorage.sol`: `registerSwappingProvider()` is public with no access control. That means that anybody can register new providers, including malicious ones. Thus, users when interacting with `transferWithTokens` and `transferWithEthers` must know what providers to pass as argument. While that does not impose a risk to StablePay, it shifts the responsibility onto users of the smart contract; if they choose malicious providers, the fault is on them. This can be mitigated by choosing providers whose owner is StablePay. **Fixed**

• `Role.sol`: `transferOwnership` is a critical method since there is no way to revert in case of a mistakenly placed incorrect address. We suggest splitting the old owner removal into a separate method as a means to temporarily allow multiple owners; then, in the case of a typo, the old owner can still react and fix ownership accordingly. **Fixed**

• `env.template`: typo KyberSwappingProvidder => KyberSwappingProvider. **Fixed**

• `ProxyBase.sol`, L66: `transferEthers` does not check whether the target address is `0x0`. We advise to do so as a means to prevent "burning" eth. **Fixed**

• `Vault.sol`, L102: `transferEthers` does not check whether the target address is `0x0`. We advise to do so as a means to prevent "burning" eth. **Fixed**

• `contracts/service/erc20/EIP20.sol`: used for test purposes only. We suggest switching to using `ERC20` instead and removing `EIP20.sol` altogether. **Fixed**

• `contracts/service/erc20/EIP20.sol`, L57: `allowance` shadows `EIP.allowance`. Consider using a different variable name. **Fixed**

• `contracts/providers/KyberSwappingProvider.sol`: `approveTokensTo(sourceToken, address(proxy), _order.sourceAmount)`; we advise resetting the approval back to zero after calling `trade(...)`, because it's possible that not all tokens will be used after the approval. **Fixed**

• `contracts/providers/KyberSwappingProvider.sol`, L321: the comment "// Execute the swapping from ERC20 token to ETH." is misleading. The swap actually happens the other way. **Fixed**

• There is an [unnecessary] inconsistency in swapping behaviour: in Kyber's `swapToken`, when calling `transferDiffTokensIfApplicable` the tokens go to StablePay first and then to the original address triggering the transaction; in Uniswap, they go directly to the order's `from` address. It is unclear that is necessary. **Fixed**

• `contracts/utils/Bytes32ArrayLib.sol`: `removeAt` as it doesn't seem to be used anywhere. **Fixed**

• There does not seem to be a way to remove a provider after it's registered, only to pause. **Fixed**

• `contracts/base/Base.sol`, L110: typo "transfer was successfully" => "transfer was successful" **Fixed**

• `contracts/interface/IVault.sol`: there're events for withdraw, but not for deposit. **Fixed**

• `contracts/interface/IVault.sol` there are events for withdraw, however, there is no function related to withdraw. **Fixed**

• `contracts/interface/IVault.sol`, L47: `deposit()` is used to store eth into the contract. It seems like the contract also can hold tokens. However, there is no interface for depositing tokens. **Fixed**

• Avoid writing functions (e.g., `checkCurrentTargetBalance`, in StablePayBase.sol, L300-315) that always revert on the case of a failure, but return `true` in case of a success (`false` is never returned). Such functions could be rewritten without returning any value. ** Partially fixed. There are still two pending instances: `transferFees` and `transferEthers`.

• `contracts/base/Vault.sol`, 34 and L46: receiving either eth or a token emits the same event: `DepositReceived`. From a single event, it is not possible to distinguish which case happened. **Fixed**

• `contracts/base/Vault.sol`, L46: comment is not enforced. The comment states "This function is used by the Base smart contract"; however, this is not enforced, anyone can deposit. **Fixed**

• `contracts/base/Vault.sol`, L87: `TokensWithdrawn` event is emitted; however, this function is performing a token transfer. Consider better aligning the event name with the function name. **Fixed**

• `contracts/base/Vault.sol`: `transferEthers` is not used anywhere. However, is a public function that is only accessible to the super user. That centralizes control to whatever funds go into the contract. **Update**: the team clarified this issue and they are ok with such centralization. **Fixed**

Best Practices to Follow (commit 1f960)

• Some dependency versions in `package.json` aren't fixed. **Fixed**

• `contracts/base/Base.sol`, L193: missing validation for the value of `to`. Check if it is different from `0x0`. **Fixed**

• Need to make sure there is at least one owner. Currently, using `deleteOwner()` is possible to arrive at a state with no owners. **Fixed**

• `contracts/base/StablePayStorage.sol`: operations on providers (`pause`/`add`/`delete`) are no longer clearly differentiated when it comes

• to pausing by admin or by provider owner. Remove this redundancy by keeping only one. **Fixed**

• `contracts/base/StablePayStorage.sol`, L400: `require(providerKey != bytes32(0x0))` is redundant; by then it's already known that provider exists. **Fixed**

• `contracts/base/Settings.sol`, L37: `value` parameter could be higher than 100. While reversible, it is a good practice to check whether the input value would be <= 100. **Fixed**

Best Practices to Follow (commit 96f39)

• `contracts/util/AddressLib.sol`: many functions state to return a boolean, but do not have a `return` statement. Examples include `requireNotEmpty`, `requireEmpty`, `requireEqualTo`, and `requireEqualTo`. **Fixed**

Appendix

File Signatures

The following are the SHA-256 hashes of the audited contracts and/or test files. A smart contract or file with a different SHA-256 hash has been modified, intentionally or otherwise, after the audit. 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 audit.

Contracts

```
3f31a9ae500ef7a2a761f895ef85e7c748d5bdbc75d7303bbbed923daed25f5a ./contracts/Migrations.sol

882796fe93c8084fb302aa9315d807cf3c4c7b1465ba92580e03b2e0f4ed86d0 ./contracts/StablePay.sol

9f57d2c76db42b98c33d90b3084cfda8c0141a3b6d8c0fe8255ce53b3017cb9b
./contracts/services/kyber/KyberNetworkProxyInterface.sol

e9e81ca33cf421385ba73c555d06f41a53fa740882eae9aece7d063acc7959b4
./contracts/services/kyber/KyberRegisterWallet.sol

4a8e07bd1b34ae0016001438b00d5f8cef22e9dddf8cb3b36ec826a376e4372a
./contracts/services/compound/ComptrollerInterface.sol

ae40c965c891e8ce4bf2045f92e9c12a059717ffe2c5c6dcd995324f4e54fbf8 ./contracts/services/compound/CErc20.sol

698377155d2dce760410a0ed4faf35b6a77a41a9edb4b2edb2933dca11872e08 ./contracts/services/compound/CToken.sol

a917454b5974986ae262b8514179ee26a2093758a1fbf5c0c3aa44fbaae33471
./contracts/services/compound/InterestRateModel.sol

17f9fff6921d8b94c3fc350bb6d74755d1e8728920d4fe7153b47679a64adb77 ./contracts/services/compound/CEther.sol

8a348e2e7af66d7a19a485f7805e699113c4656dbbf565f7681c0690bfad8478
./contracts/services/uniswap/UniswapFactoryInterface.sol

ca612a7e6c836b86d9b5b3c21281b9a78a589f15ee9ced6462c21e5af7481aff
./contracts/services/uniswap/UniswapExchangeInterface.sol

76e3688d4637fe8d81a3380f6606b45373b476671c1a41d868b6c938c5bb0469 ./contracts/services/token/WETH9.sol

b7651211badf3582db3552c1b9b3372f7a75b50f1964fd7268ab968c1896d003 ./contracts/base/Settings.sol

7110d65992efe545148d2e514d0aa7d30516c321239f2ddfcf7ea8bf32e1aa6f ./contracts/base/StablePayStorage.sol

10f5b25e1c73b68b0c9cd70174e3b03af039923e03e834213fd38c3f2a52515e ./contracts/base/Role.sol

4b890678c1b6caa909ec2877cf4da377f12810bb6a95c63e5ef2327d677e107d ./contracts/base/Vault.sol

b35e421343452a8acab5ac21cbec28349d97e4b5637e4f38a8b50ccde5bc0c00 ./contracts/base/StablePayBase.sol

fb723c760c1c151ccddbdefdb462b9903aca967c3235c18ee7588258fd995796 ./contracts/base/Storage.sol

e4d8809bb4856f39a4095aa5e5b2c4340b247b819a975eae685afe590d86e5f2 ./contracts/base/Base.sol

ab72ac5aed2a7446f25c348bf744cbf8ac54f9bf1aae5aa1123364c0a57bae72 ./contracts/base/Upgrade.sol

180f4de70797d2177be7420b41ab681cee9a047486262e734d1cb26b2efc1a8c ./contracts/base/Registration.sol

98b695a856e14287a577e5681b1aaea6e727eccce13468dc54ef03e77892d00b ./contracts/base/PostActionRegistry.sol

586ae059a87c39db8e8696ce246ec708820baa4d02a5602b0b667ba0e976d3f3 ./contracts/base/CompoundSettings.sol

d503d64918064a7ee7f09d09550859c5fb129579c3eb521ed5b0dabdc9a008ee
./contracts/base/action/EtherTransferPostAction.sol

6d3f6378bf6b163882a8de48356d4b1026808d3918045540869d1a5c35b4b9c4 ./contracts/base/action/PostActionBase.sol

9b2d303c8e13f153990924b882028246523ca6f3869b5a92df66e9ec35f749ea
./contracts/base/action/TransferToPostAction.sol

c9b6299196f1a2aa323bfa48d719465e471792677fdbeb5360b6f1db4d99049e
./contracts/base/action/CompoundMintPostAction.sol

a3a551e2db6adafaa1e3f03d3774b7aa641f9d4ea6658197e7319f6831555f38 ./contracts/base/proxy/ERCAbstractProxy.sol

d0bbf0c7802fbf78f939c6e11accddd02c50e2b3ac55282adbfaeff12da06567 ./contracts/base/proxy/IsContract.sol

2d4579547ef4b2e77a72a0d411bc435b24a86b8802363ff7c404a216a1e14e5e ./contracts/base/proxy/DelegateProxy.sol

e5c4d50dc5d0bed5b9b3a9b9eab6dbc7b26d696b9acbfe7b210293808ee64a00 ./contracts/base/proxy/ProxyBase.sol
```

fb0d0c31b65923fd1866b440f3b7a65e2eb06c77f7fc77f2efdfa0cfb29341eb ./contracts/mock/Mock.sol

a1cf313ca83d36ee0573611d79eb73ab2cd71fdc2f0aec72f3dc6fa2fc3d8ea4 ./contracts/mock/base/StablePayBaseMock.sol

bc711f49def8a9722c7d02c21e29005eb0474c81d2a2d4a09d39c647a776f9af ./contracts/mock/base/BaseMock.sol

672dde52f5aab24da8eb14a0817a1999b7ed744859614deeb9b0568760ad82a6 ./contracts/mock/base/ITransferMock.sol

204a75d6e4e52bc52f483e609b603fc9c4cf7d6b4084427ea9143f18355d9466 ./contracts/mock/base/StablePayStorageMock.sol

61da1ec0347d91e89a9ed4ba2784037fbb7cc9a457659d61f9815910fdc26602
./contracts/mock/base/action/PostActionBaseMock.sol

f1a5e015fbccc99f8ce49ae1fad9930693e912127de3aedc8994bc3db9ff922b
./contracts/mock/provider/SwappingProviderMock.sol

ac343230c68aaecacedb114ce09d7b5550856be0bd57de9fad9b50cb21615049
./contracts/mock/provider/CustomSwappingProviderMock.sol

ac70e5cb646b4cb654e669b25966d512edde01f2544a9e5c7be2f7121a7ae77c
./contracts/mock/provider/KyberSwappingProviderMock.sol

63778df87188298684abb3ae9257fb1d73b33b885e7896213caaba88b833a65d
./contracts/mock/provider/CustomUniswapExchangeMock.sol

9e9e8d0a8243cb4a6b7bbdd4020c559789b885cd04ff6ce5bcee7002fa853a76
./contracts/mock/provider/CustomUniswapFactoryMock.sol

f53eff0edef88ae4112ea8ecdbc966cca7ad7214dc9c2355f3b55907cee35aa1 ./contracts/mock/util/Bytes32ArrayMock.sol

b846d043a43c5684ac4d0ac7bfd3d640492c0d8bb6cce08a6b8f004185cd1796 ./contracts/mock/proxy/DelegateProxyMock.sol

59dcfc19674f5b77ec119a265b70d8b2b2aff296be25251467601ce1491a4fbd ./contracts/mock/proxy/ProxyBaseMock.sol

3e88aca4e230d454c9a7603c8ed45d9829b91ae5c839eedbfeffde967b5b9779 ./contracts/mock/proxy/IsContractMock.sol

a3787792c10794270e749d79919b53970bfbe18a92076239ee595838fbe2791f ./contracts/mock/proxy/ProxyTargetMock.sol

3fed1e75ed4ce24e7239137d476748a9a417dc553623454602e2c25ce0884b95 ./contracts/mock/proxy/IProxyTargetMock.sol

f5ab2a4bc0a509915731ffa82187904b36924448c26062daf9c2ed2e336578a5 ./contracts/mock/token/SimpleToken.sol

3251926598092614408 4cc9bb667060b332e581993b92fec9e28525218b10b58
./contracts/providers/UniswapSwappingProvider.sol

2e0a02635ee95649b385ede0a0608604ab1e6c298af627abd410f204e3bcc17d
./contracts/providers/AbstractSwappingProvider.sol

4e70b55106fb9a723abb62323eb58e99f5f388725082c22b510b7835ea414189 ./contracts/providers/KyberSwappingProvider.sol

013ac22ab0a98cec218dee1b2a3570788dda3d4ae3def2f122845f96b156b497 ./contracts/util/StablePayCommon.sol

31962f79cf04b43d2c7857bedfee6c421019e10d89854119f5d046af57b72164 ./contracts/util/AddressLib.sol

9938e87a24d5a1f49596f4d895dc6cdfb05ed47017eb66718cc903b8bfbe8bf4 ./contracts/util/Bytes32ArrayLib.sol

073b9cb6bd404e725730976ca1afb1c5259b3a20e7c7779f1c2887689884a943 ./contracts/interface/IPostAction.sol

009cc68b81b139c61e18c8da4ce8105d5723cb16f08cb7922d7ffe7b341edf31 ./contracts/interface/IOwnable.sol

4494ca371a5ac37d927ee7ced82f30fc9ea984428e5fcb34b8d98f7a624d6f9e ./contracts/interface/IPostActionRegistry.sol

0d9bc5abd92565ce44cbabf4a92b00e6193395d5359611ae9fef0dd74f6d542a ./contracts/interface/ISwappingProvider.sol

db76ba5333cc71896c68f3e6e09206a4e430acd2d7205cea16f0bee5b666d28b ./contracts/interface/IProviderRegistry.sol

d0aa14547b998662358c7fdbe535293ffbac0456a7af19f9d3494431867ae793 ./contracts/interface/ICompoundSettings.sol

fb4a5da753a1633b734cc670c7ab8a713de855bf3f20b236c6f288844fd931a6 ./contracts/interface/IVault.sol

ae3f69fae21cf4068a36ba2a8bf375fef72d2eff34716ea86b971b5a8b46c645 ./contracts/interface/IRole.sol

e069e94d2876f24b3b99b39428b7413e35d71fb109288d2b6ece63e7868f95f0 ./contracts/interface/IStorage.sol

329b8577793523db9d7fa2d6d27a7a48b584b3199c46d2293d384441b61bc897 ./contracts/interface/IStablePay.sol

54f98ad2f7dd08dc88f3002a00363c30cafaf95c358fe9d5ee9669bcae2af26b ./contracts/interface/IRegistration.sol

a044e32b96383ab9a81cd70b11ea4f8e2999a087a1ba4f348004be96cc19075d ./contracts/interface/ISettings.sol

76652741c11d405741f74975ce4d0cf9ee399bc27fc1e0ea466cb85a44a5f832 ./contracts/interface/IUpgrade.sol

Tests

597c604d039d78b5f3f1f3757945740d872a4fa21021d0e3632a97d334129c67 ./test/StablePayTest.js

8a2de8f3b0e7c4d2b6b4ea1e010b5b1d58934719f8053eabfe183b60979c5089
./test/base/StablePayBaseTransferTokensIfTokensAreEqualsTest.js

cfe7db35b9dd689abcfbb9f7d179df4aefe65ed8a642a9288096d84451c0d204
./test/base/StablePayStorageRegisterSwappingProviderTest.js

fa37c5eff3ee725f8a8bfc93297d0565daf6d56a8b00afd96ed2d83846b2b94b ./test/base/RoleTest.js

9adceedc2dd00703be62ccbdbe3aa4e54b43a16b56f971b95cdac4ddee1a6362
./test/base/StablePayBaseAllowanceHigherOrEqualsTest.js

2085de0df58262e4ecd891a6f2eda75940c8c44ffb15b54ba142e01e31f1b092 ./test/base/SettingsTest.js

9c4fa990fc306a0aa739a2e21afaefdda7a963c01bf6adb8ff33c9777dc19f6a ./test/base/StablePayBaseGetFeeAmountTest.js

618197134b0bc4c13d6a0d790c7839ca43b1edf21e4a187074036f0ef8ad9629 ./test/base/BaseTest.js

```
42eb1c473fbeec043da73d46634764ddd06dfb04e712b10ca9be927e2fbdd4bb ./test/base/StablePayStorageModifiersTest.js

56519fe7561c9ce764fc862ac1185305820606ad83b41759bcc0dffc38356862
./test/base/StablePayStorageGetExpectedRatesTest.js

3024f44b7042be5541ed9606c5548dbb83ce6288b98a098d7ad70f7fa893dfd0
./test/base/StablePayBaseCheckCurrentTargetBalanceTest.js

cfee22768aa982024e03657b6e3922d3a0f2cdd73924589b51691cf79f196cf5 ./test/base/UpgradeTest.js

6cff0a9f427426431479a93d458627f9dc6cd4893baaf49c58fc02e608a7976c
./test/base/StablePayBaseCalculateAndTransferFeeTest.js

5656e9386e701a1a795abb44e292870377ad925deaba4130fe78f36b44c464e4
./test/base/StablePayStorageUnpauseByAdminSwappingProviderTest.js

9962a8b50ae8c85227e2e7ac670e8c9a78812c6e358ff663c49a982afe4e571f
./test/base/StablePayStoragePauseByAdminSwappingProviderTest.js

21bc332c93f59105d361441346528bd87c03fae253a423e06c6361bbe04b6a48 ./test/base/RegistrationTest.js

f870993212a082a985330e8f09ba3c6bdbbf30bd3283c58d813ad92e0a903ea8
./test/base/StablePayStorageGetExpectedRateRangeTest.js

1a33b3612af8e24c5db0816d3ecd2315a37d7cecbf7e644034e7def651392f2c ./test/base/CompoundSettingsTest.js

843a639391b1b39a748750899d80cf53fbd869b2742871c21988b7208bc94e04 ./test/base/StorageTest.js

85741939a49d0477b7d212b77f7c51b83112e0738f97d0136e594df02b992364 ./test/base/StablePayBaseTransferFromTest.js

fe2795ffbcaaa6e1e0e85484e0653ca043d7479d5ce5ad86bae9a24722b3ace9 ./test/base/VaultTest.js

b9fdbac62fcb1a699330dabcafe62b3e9efdd185c147bb8e9547a282859cc204
./test/base/StablePayStorageUnregisterSwappingProviderTest.js

e5ada69a0abb6663f3fed4356955d8efb6fa41a392e71b01c5061814ee1f96b9 ./test/base/action/PostActionBaseTest.js

120dd21194c19dc58414645b452f41f5082342e44ce7b9338e8b1cc743b5ec51 ./test/base/proxy/DelegateProxyTest.js

20e574b9ce2d1a98f701ca4c604cc77d8ccc8d14301fbcc328d18fab6a581406 ./test/base/proxy/IsContractTest.js

7e743e7c1931e59447aa3d66489e0e19fe385de8c5873875e526871bb41df228 ./test/base/proxy/ProxyBaseTest.js

96528f204994ebfc603705fdec39773cd92e749760e3cd911fc8c8df12ed5486 ./test/factories/UniswapOrderFactory.js

08762c83fd54ec87b299454b6be8635aad4ebc9cacd6c30bd55e12327798fa50 ./test/factories/KyberOrderFactory.js

d62df1709d6cc32fb9103da2c07d1772f5b0ed07a07fbdba43f9e3c443a6c784 ./test/factories/BaseOrderFactory.js

310ccb5b49a3cd9b62902a1cae850af1ecd99ca8dbd5b4b2a0682a1393c83804
./test/providers/UniswapSwappingProviderGetExpectedRateTest.js

5af11c6481824f049a6af6560ba0d79fc48f415e9fbad0abe4650b4e0b089e67
./test/providers/SwappingProviderCalculateDiffBalanceTest.js

1b6c2282a1fc3f395e2a3e7a29636aa199b8e738777903bffe81a8276c44f6c0 ./test/providers/KyberSwappingProviderTest.js

e633f6c23798d3ebe4e762a9c9c4f8e4ece5d8f5d1ccc7945e80d9919cd9e5a2 ./test/util/events.js

d98438a62f895f16ffbe8127d7d390c4580b7de6b335821ed5371411a10e40f9 ./test/util/consts.js

2809e26a4f887513429a85e17163e0c79b29919afd5564a21bb8bfb5aa78d14a ./test/util/Bytes32ArrayLibTest.js

818f53d66fd2686984161715053265 99a4779fd2a18c18e618878a7cd810f2f4 ./test/uniswap/uniswap.js

12f14238c43dd8b9c23e13327084353af33b8d2fd254643416c0d8effcdc3e2a ./test/uniswap/deploy.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.