Quantstamp Security Assessment Certificate

November 5th 2021 — Quantstamp Verified

Hanzo

This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	DeFi protocol
Auditors	Kacper Bąk, Senior Research Engineer Souhail Mssassi, Research Engineer
Timeline	2021-10-25 through 2021-11-05
Languages	Rust
Methods	Architecture Review, Unit Testing, Manual Review
Specification	None
Documentation Quality	Low
Test Quality	Low

Source Code	
Repository	Commit
CasperLabs-UniswapV2-core	3069edd
Casperlabs-UniswapRouter	9f464a6

Total Issues	11 (0 Resolved)
High Risk Issues	4 (0 Resolved)
Medium Risk Issues	2 (0 Resolved)
Low Risk Issues	3 (0 Resolved)
Informational Risk Issues	1 (0 Resolved)
Undetermined Risk Issues	1 (0 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.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

We have reviewed the code and evaluated it with respect to the following reference implementations:

1. Uniswap V2 Core,
2. Uniswap V2 Periphery, and
3. Uniswap Flash Swapper.

We have found a number of important issues which mostly caused by deviations from the original implementations. Half of the issues are of high and medium severity. Furthermore, we believe that many of the vulnerabilities could have been caught if the project was accompanied by a high quality test suite. Therefore, we recommend addressing the issues and crafting an extensive test suite with positive and negative test cases. This task should be relatively easy since all of the reimplemented contracts have proper test suites coded in Solidity.

ID	Description	Severity	Status
QSP-1	Integer Overflow / Underflow	High	Unresolved
QSP-2	Unchecked Return Value	High	Unresolved
QSP-3	Incorrect Check for `permissioned_pair_address`	High	Unresolved
QSP-4	`remove_liquidity_cspr()` Differs from Uniswap's Implementation	High	Unresolved
QSP-5	`mint_fee()` Enables Fee Only When `fee_to` Is 0x0	Medium	Unresolved
QSP-6	`pair_for()` Does Not Sort The Tokens Before Fetching a Pair	Medium	Unresolved
QSP-7	Allowance Double-Spend Exploit	Low	Unresolved
QSP-8	Race Conditions / Front-Running	Low	Unresolved
QSP-9	Missing Verification on Some Variables	Low	Unresolved
QSP-10	`set_nonce()` Adds an Arbitrary `amount` To The Nonce	Informational	Unresolved
QSP-11	Functions `skim()` And `sync()` Are Re-entrant	Undetermined	Unresolved

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.

Findings

QSP-1 Integer Overflow / Underflow

Severity: *High Risk*

Status: Unresolved

File(s) affected: `erc20/erc20/src/erc20.rs`, `wcspr/wcspr/src/wcspr.rs`, `pair/pair/src/pair.rs`, `uniswap_v2_library/uniswap_v2_library/src/uniswap_v2_library.rs`, `flash_swapper/src/flash_swapper.rs`

Description: Integer overflow/underflow occur when an integer hits its bit-size limit. Every integer has a set range; when that range is passed, the value loops back around. A clock is a good analogy: at 11:59, the minute hand goes to 0, not 60, because 59 is the largest possible minute. Integer overflow and underflow may cause many unexpected kinds of behavior and was the core reason for the `batchOverflow` attack. Here's an example with `uint8` variables, meaning unsigned integers with a range of `0..255`.

```
```
function under_over_flow() public {
    uint8 num_players = 0;
    num_players = num_players - 1; // 0 - 1 now equals 255!
    if (num_players == 255) {
        emit LogUnderflow();   // underflow occurred
    }
    uint8 jackpot = 255;
    jackpot = jackpot + 1;   // 255 + 1 now equals 0!
    if (jackpot == 0) {
        emit LogOverflow();   // overflow occurred
    }
}
```
```

Although Rust code compiled in debug mode (and ran in tests) catches the underflows and overflows, it is unclear how future-proof production code will be against these issues.
We noticed potential underflows and/or overflows in the following:

1. `erc20/erc20/src/erc20.rs` in the function `mint()`,
2. `erc20/erc20/src/erc20.rs` in the function `transfer_from()`,
3. `erc20/erc20/src/erc20.rs` in the function `make_transfer()`,
4. `wcspr/wcspr/src/wcspr.rs` in the function `mint()`,
5. `wcspr/wcspr/src/wcspr.rs` in the function `transfer_from()`,
6. `wcspr/wcspr/src/wcspr.rs` in the function `make_transfer()`,
7. `pair/pair/src/pair.rs` in the function `mint()`,
8. `pair/pair/src/pair.rs` in the function `transfer_from()`,
9. `pair/pair/src/pair.rs` in the function `make_transfer()`,
10. `uniswap_v2_library/uniswap_v2_library/src/uniswap_v2_library.rs` in the function `get_amount_out()`,
11. `uniswap_v2_library/uniswap_v2_library/src/uniswap_v2_library.rs` in the function `get_amount_in()`,
12. `flash_swapper/src/flash_swapper.rs`, lines 264, 265, 469.

Recommendation: To protect against underflows we recommend using the function `checked_sub()`. To protect against overflows we recommend using the function `checked_add()`.

QSP-2 Unchecked Return Value

Severity: *High Risk*

Status: Unresolved

File(s) affected: `pair/pair/src/pair.rs`, `flash_swapper/src/flash_swapper.rs`, `uniswap_v2_router/uniswap_v2_router/src/transfer_helper.rs`, `uniswap_v2_router/uniswap_v2_router/src/uniswap_v2_router.rs`

Description: Most functions will return a `true` or `false` value upon success. Some functions, like `send()`, are more crucial to check than others. It's important to ensure that every necessary function is checked.
Specifically, the following code locations ignore return values:

1. `pair/pair/src/pair.rs`, lines 209, 222, 585, 590 ignore return values from `transfer()`, so a transfer may fail but `swap()` would still succeed,
2. `flash_swapper/src/flash_swapper.rs`, lines 301, 303, 510, 751, and 790,
3. `uniswap_v2_router/uniswap_v2_router/src/transfer_helper.rs`, line 16,
4. `uniswap_v2_router/uniswap_v2_router/src/uniswap_v2_router.rs`, lines 239 and L626.

Recommendation: Check return values and revert where necessary.

QSP-3 Incorrect Check for `permissioned_pair_address`

Severity: *High Risk*

Status: Unresolved

File(s) affected: `flash_swapper/src/flash_swapper.rs`

Description: The function `uniswap_v2_call()` reverts if it's called by a `permissioned_pair_address` (line 98). This is likely incorrect.

Recommendation: The function should revert if it's NOT called by `permissioned_pair_address`.

QSP-4 `remove_liquidity_cspr()` Differs from Uniswap's Implementation

Severity: *High Risk*

Status: Unresolved

File(s) affected: `uniswap_v2_router/uniswap_v2_router/src/uniswap_v2_router.rs`

Description: The function `remove_liquidity_cspr()` calls `transfer_from()` in L294, whereas the original Uniswap implementation calls `safeTransfer()`. Consequently, the original

implementation sends tokens from the contract whereas the new implementation sends tokens from the caller.

Recommendation: Use `transfer()` instead of `transfer_from()` or explain why the implementations differ.

QSP-5 `mint_fee()` Enables Fee Only When `fee_to` Is 0x0

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `pair/pair/src/pair.rs`

Description: It appears to be an error in implementation of `mint_fee()`; the fee would always be sent to 0x0.

Recommendation: Reverse the check so that the opposite happens, i.e., no fee if `fee_to` is 0x0, and fee otherwise.

QSP-6 `pair_for()` Does Not Sort The Tokens Before Fetching a Pair

Severity: *Medium Risk*

Status: Unresolved

File(s) affected: `uniswap_v2_library/uniswap_v2_library/src/uniswap_v2_library.rs`

Description: `pair_for()` does not sort the tokens before fetching a pair. Consequently, it may contradict the assumptions that Uniswap users may have.

Recommendation: Sort the tokens before returning a pair.

QSP-7 Allowance Double-Spend Exploit

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `erc20/erc20/src/erc20.rs`, `wcspr/wcspr/src/wcspr.rs`, `pair/pair/src/pair.rs`

Description: As it presently is constructed, the contract is vulnerable to the allowance double-spend exploit, as with other ERC20 tokens.

Exploit Scenario:

1. Alice allows Bob to transfer `N` amount of Alice's tokens (`N>0`) by calling the `approve()` method on `Token` smart contract (passing Bob's address and `N` as method arguments)
2. After some time, Alice decides to change from `N` to `M` (`M>0`) the number of Alice's tokens Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and `M` as method arguments
3. Bob notices Alice's second transaction before it was mined and quickly sends another transaction that calls the `transferFrom()` method to transfer `N` Alice's tokens somewhere
4. If Bob's transaction will be executed before Alice's transaction, then Bob will successfully transfer `N` Alice's tokens and will gain an ability to transfer another `M` tokens
5. Before Alice notices any irregularities, Bob calls `transferFrom()` method again, this time to transfer `M` Alice's tokens.

Recommendation: The exploit (as described above) is mitigated through use of functions that increase/decrease the allowance relative to its current value, such as `increaseAllowance()` and `decreaseAllowance()`.
Pending community agreement on an ERC standard that would protect against this exploit, we recommend that developers of applications dependent on `approve()` / `transferFrom()` should keep in mind that they have to set allowance to 0 first and verify if it was used before setting the new value. Teams who decide to wait for such a standard should make these recommendations to app developers who work with their token contract.

QSP-8 Race Conditions / Front-Running

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `pair/src/pair.rs`

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 producer attacker can take advantage of this by generating and moving transactions in a way that benefits themselves.
Specifically, in the `mint_fee()` function located in `pair/src/pair.rs#L657` , the `treasury_fee` is used in calculation for the liquidity, however these variable can also be changed from the `set_treasury_fee()` function which can cause a race condition and result a miscalculation in the liquidity.

Recommendation: Race condition issues are typically difficult to address and are of low severity. We describe it here mostly for informational purposes.

QSP-9 Missing Verification on Some Variables

Severity: *Low Risk*

Status: Unresolved

File(s) affected: `uniswap_v2_router/src/uniswap_v2_router.rs`, `uniswap_v2_core/pair/pair.rs`

Description: Some arguments are not checked for validity:

1. in the `add_liquidity()` function located in `uniswap_v2_router/src/uniswap_v2_router.rs#L45` , the `amount_a_desired` and `amount_b_desired` variables are not verified, in fact anyone can inject these parameters with the value 0.
2. in the `add_liquidity_cspr()` function located in `uniswap_v2_router/src/uniswap_v2_router.rs#L214` , the `amount_token` is not verified, in fact anyone can inject it with the value 0.

3. in the `mint_fee()` function located in `uniswap_v2_core/pair/pair.rs#L658` , we are dividing by denominator without verifying its value thus we can divide by 0.

Recommendation: Add the required checks.

QSP-10 `set_nonce()` Adds an Arbitrary `amount` To The Nonce

Severity: Informational

Status: Unresolved

File(s) affected: `erc20/erc20/src/erc20.rs`, `pair/pair/src/pair.rs`

Description: The function `set_nonce()`may add an arbitrary `amount` to the nonce. It is unclear why `amount` is needed and why nonce isn't simply incremented.

Recommendation: Increment the nonce instead of adding an arbitrary value or provide an explanation why the current functionality is needed.

QSP-11 Functions `skim()` And `sync()` Are Re-entrant

Severity: Undetermined

Status: Unresolved

File(s) affected: `pair/pair/src/pair.rs`

Description: Unlike in Uniswap, both functions are re-entrant.

Recommendation: Add a mutex to protect against re-entrancy.

Adherence to Specification

Assuming that the reference implementations act as specification, the provided implementation deviates from the specification in a few key locations (as indicated in the list of vulnerabilities).

Code Documentation

The code comes with very little inline documentation. We recommend documenting the code.

Adherence to Best Practices

1. `erc20/utils/contract-utils/src/contract_context.rs#11`: magic constant. The meaning of 2 is not immediately clear.
2. A lot of code clones, e.g., `contract-utils` in multiple locations.
3. Overall, a lot of boilerplate code and clones. We recommend improving code reusability by using macros and functions.
4. `flash_swapper/src/flash_swapper.rs` 74-88 code clones.

Test Results

Test Suite Results

All tests executed successfully. We reviewed the test suite, and given a number of important issues, we recommend expanding the test suite significantly to ensure that the code works as expected. Furthermore, a good test suite would contain both positive and negative test cases.

```
running 9 tests
test erc20_tests::test_calling_construction ... ok
test erc20_tests::test_erc20_approve ... ok
test erc20_tests::test_erc20_deploy ... ok
test erc20_tests::test_erc20_burn ... ok
test erc20_tests::test_erc20_mint ... ok
test erc20_tests::test_erc20_transfer ... ok
test erc20_tests::test_erc20_transfer_from ... ok
test erc20_tests::test_erc20_transfer_from_too_much ... ok
test erc20_tests::test_erc20_transfer_too_much ... ok

test result: ok. 9 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 5.60s

running 5 tests
test factory_tests::test_calling_construction ... ok
test factory_tests::test_factory_create_pair ... ok
test factory_tests::test_factory_deploy ... ok
test factory_tests::test_factory_set_fee_to ... ok
test factory_tests::test_factory_set_fee_to_setter ... ok

test result: ok. 5 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 11.42s

running 2 tests
test flash_swapper_tests::test_flash_swapper_deploy ... ok
test flash_swapper_tests::test_calling_construction ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 3.25s

running 12 tests
test pair_tests::test_calling_construction ... ok
test pair_tests::test_pair_approve ... ok
test pair_tests::test_pair_deploy ... ok
test pair_tests::test_pair_initialize ... ok
test pair_tests::test_pair_set_treasury_fee_percent ... ok
test pair_tests::test_pair_skim ... ok
test pair_tests::test_pair_swap ... ok
test pair_tests::test_pair_sync ... ok
test pair_tests::test_pair_transfer ... ok
test pair_tests::test_pair_transfer_from ... ok
test pair_tests::test_pair_transfer_from_too_much ... ok
```

```
test pair_tests::test_pair_transfer_too_much ... ok

test result: ok. 12 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 33.69s

running 7 tests
test wcspr_tests::test_calling_construction ... ok
test wcspr_tests::test_wcspr_approve ... ok
test wcspr_tests::test_wcspr_deploy ... ok
test wcspr_tests::test_wcspr_transfer ... ok
test wcspr_tests::test_wcspr_transfer_from ... ok
test wcspr_tests::test_wcspr_transfer_from_too_much ... ok
test wcspr_tests::test_wcspr_transfer_too_much ... ok

test result: ok. 7 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 2.76s

running 7 tests
test uniswap_v2_library_tests::test_library_deploy ... ok
test uniswap_v2_library_tests::quote ... ok
test uniswap_v2_library_tests::test_uniswap_get_amount_in ... ok
test uniswap_v2_library_tests::test_uniswap_get_amount_out ... ok
test uniswap_v2_library_tests::test_uniswap_get_amounts_in ... ok
test uniswap_v2_library_tests::test_uniswap_get_amounts_out ... ok
test uniswap_v2_library_tests::test_uniswap_get_reserves ... ok

test result: ok. 7 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 47.87s

running 13 tests
test uniswap_tests::add_liquidity_cspr ... ok
test uniswap_tests::add_liquidity ... ok
test uniswap_tests::remove_liquidity ... ok
test uniswap_tests::remove_liquidity_cspr ... ok
test uniswap_tests::remove_liquidity_cspr_with_permit ... ok
test uniswap_tests::remove_liquidity_with_permit ... ok
test uniswap_tests::swap_cspr_for_exact_tokens ... ok
test uniswap_tests::swap_exact_cspr_for_tokens ... ok
test uniswap_tests::swap_exact_tokens_for_tokens ... ok
test uniswap_tests::swap_exact_tokens_for_cspr ... ok
test uniswap_tests::swap_tokens_for_exact_tokens ... ok
test uniswap_tests::swap_tokens_for_exact_cspr ... ok
test uniswap_tests::test_uniswap_deploy ... ok

test result: ok. 13 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 104.51s
```

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

```
397ba200afde163a3799c0cfcbe888e96a6e406fe02d77006c322ad74f1023ca ./wcspr.rs
d70ae2016155f9f371a40e63ac6e56d27cf60912c6e2005db5bf8a1223a992fe ./lib.rs
6b5228543b18d216447260c0e5d62f816e9d703108e304cf1f4877e7009d03f6 ./data.rs
249c4607e37866d7be20d3eb19367726bbae950f2f13d9fbcb3a1859ff02929b ./wcspr_token.rs
efc9db18db4c5e92e9435f9cc9c423086b47525a445ff5c26ed87fd479b4b0bd ./lib.rs
5f75a1b2b7eb22eb4b35d0421445687ae5f0c2af0ba1a5cec38da3c2c3efa046 ./pair.rs
746910bc68e72d1cdcb223b42bdbe4630c2cb6eae7a70d1b812779dd303a39b5 ./data.rs
281a8de92cadb75b9a262143539b3916bc8651a528840adaafa780bca72c436a ./pair_token.rs
530620e8495b77380947115ee139cd03297535f188867dad1a0849a0b5a80cab ./lib.rs
25e09dc6ce5d1fc355733effe088e72ab9bcbe9df59a878f0562a658eaf0cf1c ./data.rs
4e281f90f6418b8b6d0571e7bdf109710a65a2a0e4733d1d51b889239cdccff1 ./flash_swapper.rs
89fd25b22b2efa35bf74fda36bbf31188635f2ccb7d1f3a76dba52c7210bddca ./flash_swapper.rs
de51b199bfd1995560153188c7ee1d0523497db106849f6cbbf5f6556eafcb9f ./factory.rs
f1b82bc5fcd9691203a9e25b17b0bba5b292d9c2414c372196086715ddff49d4 ./lib.rs
f1f25d1f0bf4526a470493dd2fa0bc89b77aeeb8d8d6622e1761e66dd13986aa ./data.rs
d1513d06730cb02a4b61abe3f8aa150f3b9f0a92927fd10cadaa407200ab6210 ./factory.rs
fb93a2da65158adeb70491ef567079783f6ab59523adf4e57cc045d563d753da ./lib.rs
98b4a6813e92dc9ac1188efa7212dcabbb8e53034f61fccc128808fbe3b33d00 ./data.rs
a0e4a95b693f16dbb5a46966aea9555cd9d7ba36c0dbc0db1d6aae00fc7818c3 ./erc20.rs
109f278b5c5291bd8629c06cc62f3937dc7dd22d4d8ccbc4955beb0c8e8a4f95 ./erc20_token.rs
ad2eb8071440a0689bff0ffcf846999fea96ada164fbd78d789a581633ee10bd ./uniswap_v2_library.rs
83e8a6df87d0af54fc792cc565006937e17d61caba457db5bf5e0e842aae7a5c ./config.rs
2376ca113557de30a76708f6ca2041431394956d0459d28c6016ffc13c2a3dd2 ./lib.rs
a78dab169d46e8cd6bd9244141fd8bc76a62ece45ba5e937eed1efccb563986e ./data.rs
9d22e374fea23deb33ce0980c311347dcba32fd6eabaa70059f2e3cacc611cbc ./uniswap.rs
472e36e42fb432cb1fd23aa4a32f650bf162274cb9a85d68d0130881ca3116e5 ./uniswap_v2_router/src/transfer_helper.rs
e58e963f92cb80e8c05999bf2b0c37f0640442777b202d434ebacb65a187cb3e ./uniswap_v2_router/src/uniswap_v2_router.rs
e3b7d90a0b2b6d1b28bb61d5e0910bd1e9293c3f7503399bbac2b6336eb3133f ./uniswap_v2_router/src/config.rs
7fe7addbc2228d1a6d5f56c310ce9b3a6e4f228363953242345e300d594fd027 ./uniswap_v2_router/src/lib.rs
9694b9ef58037ff7fe57297c15e6a50f220d4955ea8c0bc9fe55fe392c677727 ./uniswap_v2_router/src/data.rs
04dc794ec5c0745fcc84789697c8abffd034821e4707913d7fd61d71ddc569f3 ./uniswap_v2_router/bin/uniswap.rs
```

Tests

```
7b56b42bab18fdc2fe58218a75233c9570c3153d9b4c5ecbcac7a42c73a207eb ./wcspr_tests.rs
0a95978601198a382583eba2076ba5178f1428c9bde63d2e52c44123e685c56c ./lib.rs
1bc746e64bfff283220fc5b0b5a728979217554661206 9cbb07420504a418a6f ./wcspr_instance.rs
947f7333d51f129886441f245176b0b7e6c5f014f6fe323da40d32313f52c0e6 ./test_instance.rs
```

```
ab68ee77d36795b05297eff285a08cd4f1b48a46b2dcd74cace859cba6764c35 ./lib.rs
9216ffb274254ac6f88c846fa5574c208fb8703dac271b3d08175469f350c815 ./pair_instance.rs
f56099be0cf24081e329c830c75f9d77fd2dac43135767b4ded06874b3bf5c30 ./pair_tests.rs
ee7121b3a06ef0f31b8ed231b1eddced36a135c7f1b7802bf0bddcd42a7f7eec ./test.rs
986e393ffeb206a88eec060c1ebd256dfbf85fadb5276066af419a7c48cb19ce ./lib.rs
027d6a14f50b41a81e677731e907c5161189620618066748e51f13f37e12a1c0 ./data.rs
77540cc70f5f6ed57ac144a673171d1afb4f0f166e1a57a80ccc806c1441745b ./test_token.rs
7aff60fd771ff163ed5bb0f1a2f8505830d17b332a980ebea87a11c1ecbdd17f ./test_instance.rs
a76095e89d07ca3a2200c05caf05c79fdbcc7ec9a5df14b387177d781938097d ./flash_swapper_tests.rs
2eb68f1351ebb1e94739b084082559c91794f438c5c42cc2aa9d8a9251e377f9 ./lib.rs
bc7ac0717e6c168248ae1ee736dda2428be128d2e73d302bae45ae33409bfaac ./flash_swapper_instance.rs
2d8071f72e2d9d17cdba49b4d16dc2be6cddb6f8c0708e2aaccdc348e5a0104f ./factory_instance.rs
3e6c8f8592465bcb17a2b1dd42f0aaef18acb479ad662e0e2b7185b6f403fcb4 ./lib.rs
357ffaec5495dba9e966bdef5d0a4c04f35808387cc98551187f9ed8d77e03ca ./factory_tests.rs
f3b918e51d57e362531b9e3cf16ba8d47b76d16e98c3b06cc3ec40c667ec439a ./lib.rs
8fb1f14ec0586763e3b2cf81c015a348d438a4c3acb857374437d24c7e22a035 ./erc20_tests.rs
abc242a1bee1c7f9eea6caa7125ef5678352c07123a73173629caf94fa5b02d9 ./erc20_instance.rs
ae9c529ada29174c3ce9a186ad82b688ce8f36c2b507decd5eab77453cbccc40 ./build.rs
1bc17fbe996404962648bb4de45ce70b4906e526832449bd4a738e3160cb3d5b ./integration_tests.rs
3a62e090dfbd5bb31f262f556fde6f512195703c2f50699edb4756fb413ea73e ./mappings.rs
8371bcfe991bb46c5d0f6994b57c44acc6e6c61b430f4136aabf286c52f37a25 ./main.rs
0ab56c216f1bdf65320ad303c16a3da7f2d096dba9ce1a95410fc7986b0437ef ./lib.rs
1587cba9801fd1cd164ec8102f8743cd7ff68fd0015b2991d52b3a8ae53bee40 ./uniswap_v2_library_instance.rs
b36922eac2ea3d27cfed08dac95e6c4e4cb8b574a71633e9f6fbca1e9cf7d2dc ./uniswap_v2_library_tests.rs
13a71340a70e9c86a2fa89a3b055f88549e295a7b4de283d19aa490402b37e04 ./uniswap_v2_router_tests/src/uniswap_instance.rs
5e5c2e9e3f066d68170b540eeb44cc7e4ae4c16e87e3d27f587883ab4bd5c430 ./uniswap_v2_router_tests/src/lib.rs
d3fd1e986772e33f1a806444276294f45ad481795d686f168dd7fec8eb7c89da ./uniswap_v2_router_tests/src/uniswap_tests.rs
```

Changelog

- 2021-11-05 - Initial report

About Quantstamp

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

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment 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.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.