1inch Fusion

Executive Summary

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

Type	DeFi
Timeline	2025-02-24 through 2025-02-27
Language	Rust
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• https://github.com/1inch/solana-fusion #0c36848
Auditors	• Yamen Merhi Auditing Engineer • Hamed Mohammadi Auditing Engineer • Paul Clemson Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	7 Fixed: 1 Acknowledged: 6
High severity findings ⓘ	0
Medium severity findings ⓘ	3 Fixed: 1 Acknowledged: 2
Low severity findings ⓘ	2 Acknowledged: 2
Undetermined severity findings ⓘ	0
Informational findings ⓘ	2 Acknowledged: 2

Summary of Findings

1inch Fusion allows users to create trade orders, trustlessly escrow the tokens and have whitelisted market makers fill these others (supporting both complete and partial fills). The protocol also encorporates a Dutch auction mechanism with a variable exchange rate to ensure the user gets the best possible price while maintaining a fast execution time.

Overall the code is well-written, well-documented, and follows best practices. We have found few issues and we expect the 1inch team to fix them

Fix-Review Update 2025-02-27:
Repository: `https://github.com/1inch/solana-fusion` Commit: `cdc953b06aba4dc2da9c0a8da88da35b18bd0298`
The fixes were reviewed up to the referenced commit , with the `cancel_by_resolver` function and its related additions out of scope.

The 1inch team actively addressed the issues in this report by fixing them or acknowledging them.

ID	DESCRIPTION	SEVERITY	STATUS
IFS-1	**Consider Paying Fees in the Native Sol Token when the `native_dst_asset` Flag Is Set**	● Medium ⓘ	Fixed
IFS-2	**Potential Bypass of Fees Parameters in Order Creation**	● Medium ⓘ	Acknowledged
IFS-3	**Potential Surplus Fee Bypass via Inflated `estimated_dst_amount` in `get_fee_amounts()`**	● Medium ⓘ	Acknowledged
IFS-4	**Improper Slippage Handling**	● Low ⓘ	Acknowledged
IFS-5	**Consider Validating the `authority` of the `taker_src_ata` Account**	● Low ⓘ	Acknowledged
IFS-6	**Lack of Monotonicity Enforcement in Dutch Auction Rate Bumps**	● Informational ⓘ	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
IFS-7	Missing Input Validation	● Informational ⓘ	Acknowledged

Assessment Breakdown

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

> **Disclaimer**
> Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

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

- Transaction-ordering dependence
- Mishandled exceptions
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Seed collisions
- Arbitrary CPI
- Type cosplay
- Account reloading
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Missing account validation
- Appropriate ownership checks
- Arbitrary token minting
- Proper account initialization
- Proper account closing

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. 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:
   1. 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.
   2. 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, clarity, 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.

Scope

Files Included

Repo: https://github.com/1inch/solana-fusion(0c368487fd2c4383a8f080c6f38a67fb9de98183) Files: https://github.com/1inch/solana-fusion/tree/main/programs/fusion-swap

Operational Considerations

The owner of the whitelist program is expected to add Resolvers to the whitelist in order to have takers fill the orders of the users referenced as makers.

Key Actors And Their Capabilities

Whitelist Program

The owner is allowed to:

- Register Resolvers to the `Whitelist` program.
- Deregister Resolvers from the `Whitelist` program.
- Transfer the ownership of the `Whitelist` program to a new owner.

1inch Program

- Users referenced as `makers` are allowed to:
  - create orders via `create()` function.
  - cancel orders via `cancel()` function.
- Resolvers referenced as `takers` are allowed to:
  - fill orders of makers via `fill()` function.

Findings

IFS-1

Consider Paying Fees in the Native Sol Token when the `native_dst_asset` Flag Is Set

● Medium ⓘ Fixed

> **Update**
>
> Marked as "Fixed" by the client.
> Addressed in: `b617ee0538851bc1169f3153cf6fd7095066f5d9`, `cdc953b06aba4dc2da9c0a8da88da35b18bd0298`.

File(s) affected: `fusion-swap/src/lib.rs`

Description: The protocol allows the maker to set the `native_dst_asset` flag during order creation to indicate that they want to receive the native SOL token in exchange for their source tokens. The `fill()` function considers this flag when paying out to the maker; however, it does not do the same for the protocol fee. This means the protocol fee will always be in the destination token, regardless of `native_dst_asset`. This can create issues since, when a maker sets this flag, they might not provide proper values for the destination token account.

Recommendation: If `native_dst_asset` is set, consider paying both fees and maker entitlements in the native SOL amount.

IFS-2 Potential Bypass of Fees Parameters in Order Creation

● Medium ⓘ Acknowledged

> **Update**
>
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> ```
> Noted
> ```

File(s) affected: `fusion-swap/src/lib.rs`

Description: In the `Create` context of the fusion-swap program, makers are required to specify protocol fee and integrator fee parameters as well as their corresponding associated token accounts (protocol_dst_ata and integrator_dst_ata). However, the current implementation does not enforce these parameters strictly. This means that a maker can potentially create an order without setting protocol or integrator fees or by misconfiguring the protocol_dst_ata. Consequently, the maker might avoid fee deductions and receive a higher net amount than intended by the protocol fee design. Note that even if the UI enforces fee parameters, malicious makers could bypass these checks by directly interacting with the program.

While this issue might be less impactful because takers (or resolvers) must be whitelisted by the protocol and are incentivized to fill only orders with correct fee configurations, it still poses a risk. In practice, resolvers may refuse to fill orders that do not have fees applied to the correct token accounts, but the absence of on-chain enforcement or fixed values allows makers to manipulate these parameters.

Recommendation: Consider fixing the protocol fee to be set by the protocol owner, and having the integrator fee set in the UI or clearly documenting this risk in the protocol specification. Alternatively, if the protocol fees are intended to be set by the makers, implement on-chain validations that enforce a defined minimum and maximum for each fee, and ensure that the sum of `protocol_fee` and `integrator_fee` remains below a specified threshold, as outlined in IFS-8.

IFS-3 Potential Surplus Fee Bypass via Inflated `estimated_dst_amount` in `get_fee_amounts()`

● Medium ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Noted
```

File(s) affected: `fusion-swap/src/lib.rs`

Description: In the `get_fee_amounts()` function of the fusion-swap program, surplus fees are conditionally applied based on the comparison between the calculated destination amount (after applying the Dutch auction adjustments and subtracting the protocol and integrator fees) and the user-supplied `estimated_dst_amount`. Specifically, the surplus fee is added only if the adjusted destination amount exceeds the `estimated_dst_amount`.

This design allows a maker to avoid the impact of surplus fees by setting the `estimated_dst_amount` to an excessively high value relative to `min_dst_amount`. As a result, the condition to trigger the surplus fee is never met, and the protocol fee component intended to capture surplus (positive slippage) is effectively bypassed.

Recommendation: Similarly to IFS-1, consider using `estimated_dst_amount` as the definitive value that the maker is expected to receive, and reserve `min_dst_amount` solely as a slippage protection mechanism. In this approach, makers would not be able to arbitrarily inflate the `estimated_dst_amount` to avoid surplus fees, because the expected outcome would be tightly coupled to a realistic market-derived value. Fixing IFS-1 will nullify this issue.

Consider applying the `surplus_percentage` only to be applied when the user receives a surplus due to Dutch auction bumps exceeding their estimated amount—not when the surplus falls between their minimum slippage and estimated amount.

IFS-4 Improper Slippage Handling

● Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Noted. This behavior is expected.
```

File(s) affected: `fusion-swap/src/lib.rs`

Description: The protocol allows the maker to set a `min_dst_amount` during order creation. By definition, this amount represents the minimum destination tokens the maker is willing to receive in exchange for source tokens. However, the protocol currently uses this amount to calculate exchange rates via the `get_dst_amount()` function and then deducts fees from the calculated amount using the `get_fee_amounts()` function.

However, under certain conditions—particularly when the calculated fees (protocol, integrator, and surplus) exceed the bump provided by the Dutch auction—the resulting destination amount for the maker may fall below the intended `min_dst_amount`.

This behavior effectively breaks the slippage protection mechanism. Makers expect `min_dst_amount` to represent the minimum amount of destination tokens they will receive, but due to the fee calculations, they may receive less than this minimum.

Recommendation: Consider calculating the actual exchange rate using the provided `estimated_dst_amount` and then deducting fees from this calculated amount. If the result falls below the destination token amount derived from `min_dst_amount` as the exchange rate, reject the taker’s offer.

IFS-5 Consider Validating the `authority` of the `taker_src_ata` Account

● Low ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
While the validation you suggest could help prevent resolver mistakes when filling orders, we prefer
not to add it. This would require passing an additional account in every transaction or restricting
functionality by forcing the taker to always receive funds at their own address. Our current
configuration maintains flexibility while keeping transaction sizes minimal.
```

File(s) affected: `fusion-swap/src/lib.rs`

Description: The protocol does not enforce the `authority` constraint on the `taker_src_ata` account. This allows the taker to provide any account as long as the `mint` constraint is correct, which can lead to fund loss if an incorrect address is provided.

Recommendation: It is recommended to fully validate user-provided input data, even for privileged users. Ensure that the authority of `taker_src_ata` is the taker account or If the protocol intentionally allows takers to send funds to other accounts, consider making this explicit by introducing a `taker_receiver` account. Then, enforce that `taker_src_ata` 's `authority` is `taker_receiver` , which is the same approach the protocol takes for checking the validity of the `maker_dst_ata` account.

IFS-6
Lack of Monotonicity Enforcement in Dutch Auction Rate Bumps

● Informational ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Noted. Same as IFS-3 and IFS-4.
```

File(s) affected: `fusion-swap/src/dutch_auction.rs`

Description: The protocol documentation specifies that the auction operates on a Dutch auction model, where the price is expected to decline over time. However, in the `calculate_rate_bump()` function in `fusion-swap/src/dutch_auction.rs` , there is no check to ensure that the rate bumps specified by the user in `points_and_time_deltas` are in a strictly declining order. This omission allows a user to supply rate bump values that could be non-monotonic (i.e., a subsequent bump could be higher than a previous one), which contradicts the intended Dutch auction behavior and the protocol documentation.

Recommendation: Consider adding a check that validates the sequence of rate bumps in `points_and_time_deltas` to ensure that each subsequent rate bump is less than or equal to the preceding one.

IFS-7 Missing Input Validation

● Informational ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Noted. Same as IFS-3, IFS-4, and IFS-6.
```

File(s) affected: `fusion-swap/src/lib.rs`

Description: The fusion-swap program currently allows makers to supply fee parameters via the `ReducedFeeConfig` and rate bump values in the Dutch auction data without performing explicit bound checks. Unlike the surplus fee—which is validated to be less than `BASE_1E2` —there is no on-chain validation to ensure that:

- The protocol fee and integrator fee values (or their sum) are within a reasonable range (e.g., less than or equal to `BASE_1E5`). Without this check, makers could specify arbitrarily high fees, potentially leading to unintended pricing behavior and potential underflow when substracting the fees from the `dst_amount` .
- The rate bumps provided in the Dutch auction parameters are within an acceptable range relative to `BASE_1E5` . This extra validation would help prevent overflow in the `calculate_rate_bump()` function and ensure that the auction price adjustment behaves as intended.

Recommendation: Implement comprehensive bound checks on user-supplied inputs.

Auditor Suggestions

S1 Consider Using `UncheckedAccount<'info>` Instead of `Accountinfo<'info>`

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `28b80991a9de703bdd52153429c54b0a7550f8c8`.

File(s) affected: `fusion-swap/src/lib.rs`

Description: The program currently uses `AccountInfo<'info>` for referencing accounts that don't require additional checks. However, it is recommended to use `UncheckedAccount<'info>`, a more recent alternative designed for this purpose.

S2 Ensure `taker_src_ata` Account Exists

Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Similar to IFS-5, your suggestion would make things easier for resolvers but would also increase
compute unit usage on every fill call. Adopting your recommendation would negatively impact a
frequently used case to improve a rarely encountered one.
```

File(s) affected: `fusion-swap/src/lib.rs`

Description: The protocol transfers the source token from the maker to the `taker_src_ata` account. However, there are no checks to ensure that this account actually exists.

Recommendation: Although takers are whitelisted entities, it is still recommended to use the `init_if_needed` account constraint to ensure the protocol functions properly.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – 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.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **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).

Test Suite Results

All tests are passing, with 60 tests executed successfully. The test data was gathered by running `anchor test`.

Fix Review Update: The tests increased with 75 tests executed successfully.

```
  Dutch Auction
    ✔ should not work after the expiration time
    ✔ should fill with initialRateBump before auction started
    ✔ should fill with another price after auction started, but before first point
    ✔ should fill with another price after between points
    ✔ should fill with default price after auction finished (801ms)
    ✔ Execute the trade with surplus
    ✔ Execute the trade with all fees (43ms)
  Fusion Swap
    Single escrow
      ✔ Execute the trade (420ms)
      ✔ Execute the trade with different maker's receiver (842ms)
      ✔ Execute the trade without u64 overflow (2478ms)
      ✔ Execute the trade with different taker's receiver wallet (433ms)
      ✔ Doesn't execute the trade when maker's token account belongs to wrong mint
      ✔ Execute the trade with native tokens => tokens (1207ms)
      ✔ Execute the trade with tokens => native tokens (1256ms)
      ✔ Execute the trade with protocol fee (835ms)
      ✔ Execute the trade with integrator fee (832ms)
      ✔ Doesn't execute the trade with exchange amount more than escow has (x_token)
      ✔ Check that maker's yToken account is created automatically if it wasn't initialized before
(1252ms)
      ✔ Fails to create with zero src amount
      ✔ Fails to create with zero min dst amount
      ✔ Fails to create if escrow has been created already (433ms)
      ✔ Doesn't execute the trade with the wrong order_id
      ✔ Doesn't execute the trade with the wrong escrow ata
      ✔ Doesn't execute the trade with the wrong dstMint
      ✔ Doesn't execute the trade with the wrong maker receiver
      ✔ Doesn't create escrow with the wrong surplus param
      ✔ Doesn't create escrow with protocol_dst_ata from different mint
      ✔ Doesn't create escrow with intergrator_dst_ata from different mint
      ✔ Doesn't execute the trade with the wrong protocol_dst_ata (442ms)
      ✔ Doesn't execute the trade without protocol_dst_ata (435ms)
      ✔ Doesn't execute the trade with the wrong integrator_dst_ata (435ms)
      ✔ Doesn't execute the trade without integrator_dst_ata (436ms)
      ✔ Execute the multiple trades (838ms)
      ✔ Execute the multiple trades, rounding (1650ms)
      ✔ Cancel the trade (417ms)
      ✔ Cancel the trade with native tokens (1255ms)
      ✔ Doesn't cancel the trade with the wrong order_id
      ✔ Doesn't cancel the trade with the wrong escrow ata
      ✔ Doesn't cancel the trade with the wrong maker
      ✔ Fails when taker isn't whitelisted (474ms)
      ✔ Execute the partial fill and close escow after (1281ms)
      ✔ Execute the trade with native tokens (SOL) as destination (1335ms)
      ✔ Fails to execute the trade if maker_dst_ata is missing (868ms)
      ✔ Fails to create if native_dst_asset = true but mint is different from native mint (42ms)
      ✔ Execute the trade and transfer wSOL if native_dst_asset = false and native dst mint is provided
(1257ms)
      Token 2022
        ✔ Execute trade with SPL Token -> Token 2022 (834ms)
        ✔ Execute trade with Token 2022 -> SPL Token (849ms)
        ✔ Execute trade between two Token 2022 tokens (838ms)
        ✔ Cancel escrow with Token 2022 (850ms)
    Optional tests
      ✔ Doesn't execute the trade with the wrong maker's ata
      ✔ Doesn't execute the trade with the wrong token
    Multiple escrows
      ✔ Double fill (1657ms)

  Whitelist
    ✔ Can register and deregister a user from whitelist (811ms)
    ✔ Cannot register the same user twice (825ms)
    ✔ Can transfer ownership to new owner (413ms)
    ✔ New owner can register and deregister users (825ms)
    ✔ Cannot register with wrong owner
    ✔ Cannot deregister with wrong owner (803ms)
    ✔ Previous owner cannot register or deregister users (457ms)
    ✔ Non-owner cannot transfer ownership


  60 passing (1m)

**Fix Review Update:**
```

```
Cancel by Resolver
    ✔ Resolver can cancel the order for free at the beginning of the auction
    ✔ Resolver can cancel the order at different points in the order time frame (162ms)
    ✔ Resolver can cancel the order after auction
    ✔ Resolver can't cancel if the order has not expired
    ✔ Resolver can't cancel if the caller is not a whitelisted resolver
    ✔ Maker can't create an escrow if the fee is greater than lamports balance

  Dutch Auction
    ✔ should not work after the expiration time
    ✔ should fill with initialRateBump before auction started
    ✔ should fill with another price after auction started, but before first point
    ✔ should fill with another price after between points
    ✔ should fill with default price after auction finished
    ✔ Execute the trade with surplus
    ✔ Execute the trade with all fees

  Fusion Swap
    Single escrow
      ✔ Execute the trade (473ms)
      ✔ Execute the trade with different maker's receiver (941ms)
      ✔ Execute the trade without u64 overflow (2831ms)
      ✔ Execute the trade with different taker's receiver wallet (472ms)
      ✔ Doesn't execute the trade when maker's token account belongs to wrong mint
      ✔ Execute the trade with native tokens => tokens (1407ms)
      ✔ Execute the trade with tokens => native tokens (1414ms)
      ✔ Execute the trade with protocol fee (951ms)
      ✔ Execute the trade with native tokens (SOL) as destination + protocol fee (1435ms)
      ✔ Execute the trade with integrator fee (950ms)
      ✔ Execute the trade with native tokens (SOL) as destination + integrator fee (1414ms)
      ✔ Execute the trade with wrapped native tokens (wSOL) as destination + protocol & integrator fee
(1428ms)
      ✔ Doesn't execute the trade with exchange amount more than escow has (src token)
      ✔ Doesn't execute the trade without taking dst ata
      ✔ Check that maker's yToken account is created automatically if it wasn't initialized before
(1420ms)
      ✔ Fails to create with zero src amount
      ✔ Fails to create with zero min dst amount
      ✔ Fails to create if escrow has been created already (485ms)
      ✔ Doesn't execute the trade with the wrong order_id
      ✔ Doesn't execute the trade with the wrong escrow ata
      ✔ Doesn't execute the trade with the wrong dstMint
      ✔ Doesn't execute the trade with the wrong maker receiver
      ✔ Doesn't create escrow with the wrong surplus param
      ✔ Doesn't create escrow with protocol_dst_acc from different mint
      ✔ Doesn't create escrow with intergrator_dst_acc from different mint
      ✔ Doesn't execute the trade with wrong protocol_dst_acc authority (485ms)
      ✔ Doesn't execute the trade with the wrong protocol_dst_acc mint (491ms)
      ✔ Doesn't execute the trade without protocol_dst_acc (487ms)
      ✔ Doesn't execute the trade with the wrong integrator_dst_acc authority (489ms)
      ✔ Doesn't execute the trade with the wrong integrator_dst_acc mint (477ms)
      ✔ Doesn't execute the trade without integrator_dst_acc (488ms)
      ✔ Execute the multiple trades (952ms)
      ✔ Execute the multiple trades, rounding (1897ms)
      ✔ Cancel the trade (475ms)
      ✔ Cancel the trade with native tokens (1435ms)
      ✔ Doesn't cancel the trade with the wrong order_id
      ✔ Doesn't cancel the trade with the wrong escrow ata
      ✔ Doesn't cancel the trade with the wrong maker
      ✔ Fails when taker isn't whitelisted (509ms)
      ✔ Execute the partial fill and close escow after (1435ms)
      ✔ Execute the trade with native tokens (SOL) as destination (1424ms)
      ✔ Fails to execute the trade if maker_dst_acc is missing (963ms)
      ✔ Fails to create if native_dst_asset = true but mint is different from native mint
      ✔ Execute the trade and transfer wSOL if native_dst_asset = false and native dst mint is provided
(1430ms)
      Token 2022
        ✔ Execute trade with SPL Token -> Token 2022 (969ms)
        ✔ Execute trade with Token 2022 -> SPL Token (965ms)
        ✔ Execute trade between two Token 2022 tokens (964ms)
        ✔ Cancel escrow with Token 2022 (949ms)
    Optional tests
```

```
      ✔ Doesn't execute the trade with the wrong maker's ata
      ✔ Doesn't execute the trade with the wrong token
      ✔ Double fill (1902ms)
      ✔ Print bumps (476ms)
      ✔ Calculate and print tx cost (4823ms)
      ✔ Calculate and print tx cost (lookup tables) (33781ms)

  Whitelist
    ✔ Can register and deregister a user from whitelist (897ms)
    ✔ Cannot register the same user twice (950ms)
    ✔ Can transfer ownership to new owner (476ms)
    ✔ New owner can register and deregister users (953ms)
    ✔ Cannot register with wrong owner
    ✔ Cannot deregister with wrong owner (933ms)
    ✔ Previous owner cannot register or deregister users (479ms)
    ✔ Non-owner cannot transfer ownership


  75 passing (2m)
```

Changelog

- 2025-02-27 - Initial report
- 2025-03-24 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by Quantstamp; however, Quantstamp does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication or other making available of the report to you by Quantstamp.

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and and may not be represented as such. No third party is entitled to rely on the report in any any way, including for the purpose of making any decisions to buy or sell a product, product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or or any open source or third-party software, code, libraries, materials, or information to, to, called by, referenced by or accessible through the report, its content, or any related related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

Quantstamp