Solana Fusion Secure Code Review

Technical Report

1inch Limited

22 May 2025
Version: 1.1

Kudelski Security – Nagravision Sàrl

Corporate Headquarters
Kudelski Security – Nagravision Sàrl
Route de Genève, 22-24
1033 Cheseaux sur Lausanne
Switzerland

Confidential

DOCUMENT PROPERTIES

Version:	1.1
File Name:	Kudelski_Security_1inch_Secure_Code_Review_v1.1.docx
Publication Date:	22 May 2025
Confidentiality Level:	Confidential
Document Owner:	Jamshed Memon
Document Authors:	Jamshed Memon
Document Recipients:	Gleb Alekseev
Document Status:	Proposal

Copyright Notice

Kudelski Security, a business unit of Nagravision Sàrl, is a member of the Kudelski Group of Companies. This document is the intellectual property of Kudelski Security and contains confidential and privileged information. The reproduction, modification, or communication to third parties (or to other than the addressee) of any part of this document is strictly prohibited without the prior written consent from Nagravision Sàrl.

TABLE OF CONTENTS

EXECUTIVE SUMMARY 4
1. PROJECT SUMMARY 5
1.1 Context 5
1.2 Scope 5
1.3 Remarks 5
1.4 Additional Note 6
1.5 Follow-up 6
2. TECHNICAL DETAILS OF SECURITY FINDINGS 7
2.1 KS-1IN-F-01 Potential Dust Order and Cancellation Attack 8
2.2 KS-1IN-F-02 Auction Duration Not Validated 10
2.3 KS-1IN-F-03 Lack of Emergency Mechanism 11
2.4 KS-1IN-F-04 Order Hash Collision 12
2.5 KS-1IN-F-05 Partial Fill Safeguards Not Sufficient 13
3. OBSERVATIONS 14
3.1 KS-1IN-O-01 Best Secure Code Practice 15
3.2 KS-1IN-O-02 Lack of Unit Test Vectors 16
3.3 KS-1IN-O-03 Fee Upper Bound Not Explicitly Set 17
3.4 KS-1IN-O-04 Token 2022 Support 18
3.5 KS-1IN-O-05 Token Program May Be Confused 19
4. METHODOLOGY 20
4.1 Kickoff 20
4.2 Ramp-up 20
4.3 Review 20
4.4 Reporting 21
4.5 Verify 21
5. VULNERABILITY SCORING SYSTEM 22
6. CONCLUSION 24
7. REFERENCES 25
DOCUMENT RECIPIENTS 26
KUDELSKI SECURITY CONTACTS 26
DOCUMENT HISTORY 26

EXECUTIVE SUMMARY

1inch Limited (“the Client”) engaged Kudelski Security (“Kudelski”, “We”) to perform the Solana Fusion Secure Code Review.

The assessment was conducted remotely by the Kudelski Security Team.

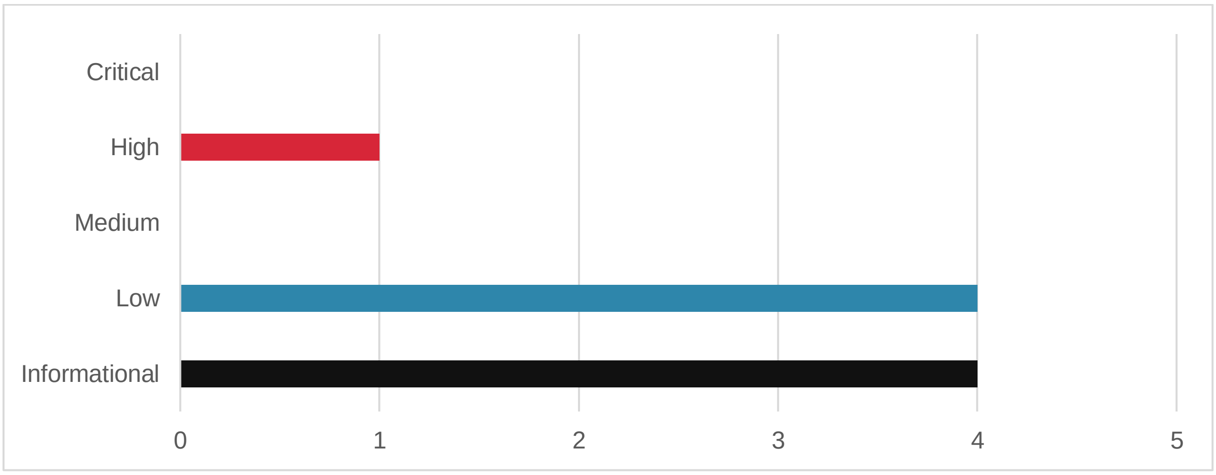
The review took place between 24 April 2025 and 02 May 2025, and focused on the following objectives:

- Provide the customer with an assessment of their overall security posture and any risks that were discovered.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the result of our tests.

Key Findings

The following are the major themes and issues identified during the audit period. These, along with other items within the findings section, should be prioritized for remediation to reduce to the risk they pose.

- Potential Dust Order and Cancellation Attack
- Lack of Emergency Mechanism



Findings ranked by severity

1. PROJECT SUMMARY

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

1.1 Context

The `solana-fusion` repository implements a protocol that brings a MEV-resistant swapping mechanism to the decentralized finance (DeFi) ecosystem based on users' signed intents.

1.2 Scope

The scope consisted in specific Rust files and folders located at:

- https://github.com/1inch/solana-fusion

  (commit: ac389333c9d7a2509de59389ac853f1420c89bf1)

The folders and files in scope are:

```
programs/fusion-swap/src/
├──── dutch_auction.rs
├──── error.rs
└──── lib.rs
programs/whitelist/src/
├──── error.rs
└──── lib.rs
```

The goal of the evaluation was to perform a security audit on the source code.

- No additional systems or resources were in scope for this assessment.
- The dependencies are out of scope of the review.
- Test codes are out of scope.

1.3 Remarks

During the code review, the following positive observations were noted regarding the scope of the engagement:

- The code is well structured.
- Quick and open communication via Telegram
- The developers have made a careful and in-depth analysis of their project.
- We had regular and enriching technical exchanges on various topics.

1.4 Additional Note

It is important to notice that, although we did our best in our analysis, no code audit assessment is per se guarantee of absence of vulnerabilities. Our effort was constrained by resource and time limits, along with the scope of the agreement.

In assessing the severity of some of the findings we identified, we kept in mind both the ease of exploitability and the potential damage caused by an exploit.

While assessing the severity of the findings, we considered the impact, ease of exploitability, and the probability of attack. This is a solid baseline for severity determination. Information about the severity ratings can be found in Chapter Vulnerability Scoring System of this document.

1.5 Follow-up

After the initial report (V1.0) was delivered, the Client acknowledged all vulnerabilities and weaknesses in the following codebase revision:

- commit: 6879deaa49cabd8c7aac3238ec03b986f4273dc0

2. TECHNICAL DETAILS OF SECURITY FINDINGS

This chapter provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the security findings.

#	SEVERITY	TITLE	STATUS
KS-1IN-F-01	High	Potential Dust Order and Cancellation Attack	Acknowledged
KS-1IN-F-02	Low	Auction Duration Not Validated	Acknowledged
KS-1IN-F-03	Low	Lack of Emergency Mechanism	Acknowledged
KS-1IN-F-04	Low	Order Hash Collision	Acknowledged
KS-1IN-F-05	Low	Partial Fill Safeguards Not Sufficient	Acknowledged

Findings overview.

2.1 KS-1IN-F-01 Potential Dust Order and Cancellation Attack

Severity	Impact	Likelihood	Status
High	High	High	Acknowledged

Description

The protocol allows creation of arbitrarily small orders without minimum size requirements. A malicious user can create multiple small orders that are uneconomical for resolvers to fill due to transaction costs. Number of such orders can overwhelm the system.

Impact

1. Resolvers lose money filling dust orders due to transaction costs exceeding profit
2. System spam through creation/cancellation of dust orders
3. Denial of service for legitimate orders as resolvers must filter through spam
4. Economic drain on resolvers who attempt to fill dust orders

Evidence

```
    pub fn create(ctx: Context<Create>, order: OrderConfig) -> Result<()> {
    require!(
        order.src_amount != 0 && order.min_dst_amount != 0,
        FusionError::InvalidAmount
    );
    // Only checks for non-zero, no minimum size requirement
}

pub fn cancel(ctx: Context<Cancel>, order_hash
: [u8; 32], order_src_asset_is_native: bool) -> Result<()> {
    // Returns all tokens
    transfer_checked(
        CpiContext::new_with_signer(
            ctx.accounts.src_token_program.to_account_info(),
            TransferChecked {
                from: ctx.accounts.escrow_src_ata.to_account_info(),
                // ... other fields ...
            },
        ),
        ctx.accounts.escrow_src_ata.amount,  // Returns full amount
        ctx.accounts.src_mint.decimals,
    )?;

    // Returns all rent
    close_account(CpiContext::new_with_signer(
        ctx.accounts.src_token_program.to_account_info(),
        CloseAccount {
            account: ctx.accounts.escrow_src_ata.to_account_info(),
            destination: ctx.accounts.maker.to_account_info(),  // Returns rent
            // ... other fields ...
        },
    ))
}
```

```
Location: programs/fusion-swap/src/lib.rs, line 44, 301
```

Affected Resources

- `programs/fusion-swap/src/lib.rs`

Recommendation

Add minimum order size requirement. E.g. minimum size of Jupiter limit order is $5.

Reference

None

2.2 KS-1IN-F-02 Auction Duration Not Validated

Severity	Impact	Likelihood	Status
Low	Medium	Low	Acknowledged

Description

In the current implementation of Dutch auctions, there is no validation for the auction duration. The `calculate_rate_bump` function does not validate duration auctions, which can bypass the intended price decay mechanism.

Impact

By setting `duration` with a very small value (for example, zero), fillers can execute trades immediately at the initial price, which could lead to potential MEV opportunities for fillers. As a result, makers may receive unfavourable execution prices.

Evidence

```
pub fn calculate_rate_bump(timestamp: u64, data: &AuctionData) -> u64 {
    if timestamp <= data.start_time as u64 {
        return data.initial_rate_bump as u64;
    }
    let auction_finish_time = data.start_time as u64 + data.duration as u64;
    if timestamp >= auction_finish_time {
        return 0;

    }
```

`Location: programs/fusion-swap/src/auction.rs, line 17`

Affected Resources

- `programs/fusion-swap/src/auction.rs`

Recommendation

The Client commented that "*Our backend validates all orders before passing them to resolvers. Orders with invalid parameters, including the auction, will not pass validation*."

The security measures on backend are out of scope. Hence, we still recommend to validate the auction duration in the order creation and ensure the auction duration is longer than the minimum value.

Reference

None

2.3 KS-1IN-F-03 Lack of Emergency Mechanism

Severity	Impact	Likelihood	Status
Low	Low	Low	Acknowledged

Description

The protocol lacks a way to pause operations if critical vulnerabilities are found or during market anomalies. Once the `fill` function is called, trades cannot be stopped or reversed. In the event of a serious security issue, an emergency shutdown or a pause mechanism would be useful to halt all transactions and prevent additional damage immediately.

Impact

Lack of emergency function could lead to a severe damage to the business. An unexpected security breach may cause tokens from the escrow's account to the hacker's account, however, such attack may not be stopped immediately due to the lack of shutdown mechanism.

Evidence

```
  pub fn fill(ctx: Context<Fill>, order: OrderConfig, amount: u64) -> Result<()> {
...
        // Escrow => Taker
        transfer_checked(
            CpiContext::new_with_signer( ... ),
            amount,
            ctx.accounts.src_mint.decimals,
        )?;
```

Location: `programs/fusion-swap/src/lib.rs, line 166`

Affected Resources

- `programs/fusion-swap/src/lib.rs`

Recommendation

Create an emergency state and check whether the state is enabled or not before the fill function is called.

Reference

- https://docs.makerdao.com/smart-contract-modules/shutdown

2.4 KS-1IN-F-04 Order Hash Collision

Severity	Impact	Likelihood	Status
Low	Low	Low	Acknowledged

Description

The `order_hash` function uses only the serialized `OrderConfig`, plus optional destination accounts, mint addresses, and a receiver key. A malicious user could recreate an identical order with the same parameters (i.e., `id, src_amount, min_dst_amount, fee configuration`, etc.) to produce the same hash. This can lead to potential collisions in escrow accounts and confusion in order tracking.

Impact

Multiple orders from the same maker may map to the same PDA if they reuse identical parameters. So, collisions could disrupt order management and record keeping.

Evidence

```
  fn order_hash(
    order: &OrderConfig,
    protocol_dst_acc: Option<Pubkey>,
    integrator_dst_acc: Option<Pubkey>,
    src_mint: Pubkey,
    dst_mint: Pubkey,
    receiver: Pubkey,
) -> Result<[u8; 32]> {
    Ok(hashv(&[
                // ... other code ...
    ])
    .to_bytes()) }
```

`Location: programs/fusion-swap/src/lib.rs, line 732, 745`

Affected Resources

- `programs/fusion-swap/src/lib.rs`

Recommendation

Add a nonce or timestamp field in `OrderConfig` to ensure every created order is hashed differently. Concatenate the nonce/timestamp with existing parameters when computing `order_hash`.

Reference

None

2.5 KS-1IN-F-05 Partial Fill Safeguards Not Sufficient

Severity	Impact	Likelihood	Status
Low	Medium	Low	Acknowledged

Description

The current implementation calculates the price per partial fill, so each fill can have different rates and fee adjustments. As a result, market changes between fills and the ongoing Dutch auction adjustments mean that there is no strict guarantee of the absolute best execution price when filling in multiple partial transactions.

Impact

The maker may lose the profit from the partial fills, compared to the one-off swap, due to the market price manipulation attack or the sandwich attack.

Evidence

The partial fill price is calculated as below:

```
        let dst_amount = get_dst_amount(
            order.src_amount,
            order.min_dst_amount,
            amount,
            Some(&order.dutch_auction_data),
        )?;
```

`Location: programs/fusion-swap/src/lib.rs, line 186`

Then, the price is adjusted on time by Dutch auction.

```
if let Some(data) = opt_data {
    let rate_bump = calculate_rate_bump(Clock::get()?.unix_timestamp as u64,
data);
    result = result
        .mul_div_ceil(BASE_1E5 + rate_bump, BASE_1E5)
        .ok_or(ProgramError::ArithmeticOverflow)?;
}
```

`Location: programs/fusion-swap/src/lib.rs, line 775`

Affected Resources

- `programs/fusion-swap/src/lib.rs`

Recommendation

Add restrictions on minimum fill size or maximum number of partial fills. Also, add price impact limits between fills by introducing a real-time price feed or oracle for each partial fill, ensuring consistent and fair pricing.

Reference

None

3. OBSERVATIONS

This chapter contains additional observations that are not directly related to the security of the code, and as such have no severity rating or remediation status summary. These observations are either minor remarks regarding good practice or design choices or related to implementation and performance. These items do not need to be remediated for what concerns security, but where applicable we include recommendations.

#	SEVERITY	TITLE	STATUS
KS-1IN-O-01	Informational	Best Secure Code Practice	Informational
KS-1IN-O-02	Informational	Lack of Unit Test Vectors	Informational
KS-1IN-O-03	Informational	Fee Upper Bound Not Explicitly Set	Informational
KS-1IN-O-04	Informational	Token 2022 Support	Informational
KS-1IN-O-05	Informational	Token Program May Be Confused	Informational

Observations overview.

3.1 KS-1IN-O-01 Best Secure Code Practice

Description

- Wrong Error Type: `InconsistentProtocolFeeConfig` is used instead of `InconsistentIntegratorFeeConfig` for integrator validation.

```
if integrator_fee_amount > 0 {
...
    .ok_or(FusionError::InconsistentProtocolFeeConfig)?
```

Location: programs/fusion-swap/src/lib.rs, line 258

Affected Resources

- `programs`

Recommendation

Update the in-line comments properly, apply the type conversion consistently, and improve the code readability.

Reference

None

3.2 KS-1IN-O-02 Lack of Unit Test Vectors

Description

According to the `cargo llvm-cov` tool (v0.6.10), the overall test coverage of code in scope reaches less than 4%. If the test coverage is too low, hidden vulnerabilities may be introduced without being detected when the code base is updated.

Evidence

FILENAME	FUNCTION COVERAGE	LINE COVERAGE	REGION COVERAGE	BRANCH COVERAGE
fusion-swap/src/auction.rs	0.00% (0/6)	0.00% (0/48)	0.00% (0/22)	- (0/0)
fusion-swap/src/error.rs	0.00% (0/1)	0.00% (0/1)	0.00% (0/1)	- (0/0)
fusion-swap/src/lib.rs	3.03% (1/33)	0.23% (1/441)	0.50% (1/200)	- (0/0)
whitelist/src/error.rs	0.00% (0/1)	0.00% (0/1)	0.00% (0/1)	- (0/0)
whitelist/src/lib.rs	5.26% (1/19)	3.12% (1/32)	5.26% (1/19)	- (0/0)
Totals	**3.33% (2/60)**	**0.38% (2/523)**	**0.82% (2/243)**	**- (0/0)**

Affected Resources

- `solana-fusion`

Recommendation

Add more test vectors and increase the unit test coverage.

Reference

None

3.3 KS-1IN-O-03 Fee Upper Bound Not Explicitly Set

Description

In `FeeConfig`, both the protocol fee and the integrator fee are defined as `u16` without maximum limits, which means that both fees are implicitly bounded to 65535. However, it is not clear this is the intended upper bound of fees.

Protocol or integrators could set fees higher than 100%. Then, combined fees could exceed the transaction amount and makers could lose more in fees than intended.

Evidence

```
fn get_fee_amounts(
    integrator_fee: u16,
    protocol_fee: u16,
    surplus_percentage: u8,
    dst_amount: u64,
    estimated_dst_amount: u64,
) -> Result<(u64, u64, u64)> {
    let integrator_fee_amount = dst_amount
        .mul_div_floor(integrator_fee as u64, BASE_1E5)
        .ok_or(ProgramError::ArithmeticOverflow)?;

    let mut protocol_fee_amount = dst_amount
        .mul_div_floor(protocol_fee as u64, BASE_1E5)
        .ok_or(ProgramError::ArithmeticOverflow)?;
```

`Location: programs/fusion-swap/src/lib.rs, line 84`

Affected Resources

- `programs/fusion-swap/src/lib.rs`

Recommendation

The Client commented that “This upper limit is known and intended.” Based on this comment, we recommend to add the in-line comment to the code and the document to explain the fee limit, avoiding confusion.

Based on the client's confirmation that "This upper limit is known and intended", we recommend to add inline comment to the code to explain the fee limits and update the protocol document.

Reference

None

3.4 KS-1IN-O-04 Token 2022 Support

Description

The fee calculation doesn't account for Token-2022 transfer fees. Also, ATAs are created without validating the correct token program.

Evidence

```
fn get_fee_amounts(
    integrator_fee: u16,
    protocol_fee: u16,
    surplus_percentage: u8,
    dst_amount: u64,
    estimated_dst_amount: u64,
) -> Result<(u64, u64, u64)> {
...
   /// Maker's ATA of dst_mint
    #[account(
        init_if_needed,
        payer = taker,
        associated_token::mint = dst_mint,
        associated_token::authority = maker_receiver,
        associated_token::token_program = dst_token_program,
```

`Location: programs/fusion-swap/src/lib.rs, line 84, 577`

Affected Resources

- `programs/fusion-swap/src/lib.rs`

Recommendation

Handle the token-2022 transfer fees if `dst_token_program` is the token 2022. Also, add validation for ATA program to ensure ATA matches with `dst_token_program`.

Reference

None

3.5 KS-1IN-O-05 Token Program May Be Confused

Description

In the `fill` and `create` instructions, the token program is not validated against the mint's owner. This could allow an attacker to use a mismatched token program, potentially bypassing expected behaviour or introducing vulnerabilities. If the mint belongs to Token-2022 but the program is set to the SPL Token program, these features might not be handled correctly.

If the token program (`src_token_program` or `dst_token_program`) does not match the mint's owner, the protocol could interact with an unintended token program. This could lead to unexpected behaviour, such as incorrect fee calculations or unauthorized token transfers.

Evidence

```
    pub struct Fill<'info> {
...
    /// Maker asset
    src_mint: Box<InterfaceAccount<'info, Mint>>,
    /// Taker asset

    dst_mint: Box<InterfaceAccount<'info, Mint>>,

...

    src_token_program: Interface<'info, TokenInterface>,

    dst_token_program: Interface<'info, TokenInterface>,
```

`Location: programs/fusion-swap/src/lib.rs, line 17`

Affected Resources

- `programs/fusion-swap/src/lib.rs`

Recommendation

Add constraints to ensure that the token program matches the mint's owner. For example, the following pseudo code might be used.

```
    pub struct Fill<'info> {
    #[account(
        constraint = src_mint.to_account_info().owner == src_token_program.key()
        constraint = dst_mint.to_account_info().owner == dst_token_program.key()
...
    #[account(
        constraint = src_token_program.key() == dst_token_program.key()
```

Reference

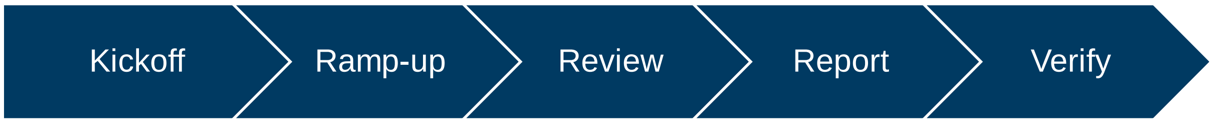
None

Client’s Comment

Our understanding was as follows: `maker_dst_ata` belongs to both `dst_mint` and `dst_token_program` since we specify this in the constraints. Thus, this account cannot be created by the associated token program if the mint and token_program don't match. Based on this comment, this finding was moved to the Observations.

4. METHODOLOGY

For this engagement, Kudelski Security used a methodology that is described at a high level in this chapter. This is broken up into the following phases.



4.1 Kickoff

The Kudelski Security Team set up a kickoff meeting where project stakeholders were gathered to discuss the project as well as the responsibilities of participants. During this meeting, we verified the scope of the engagement and discussed the project activities.

4.2 Ramp-up

Ramp-up consisted of the activities necessary to gain proficiency on the particular project. This included the steps required for gaining familiarity with the codebase and technological innovations utilized.

4.3 Review

The review phase is where most of the work on the engagement was performed. In this phase we have analyzed the project for flaws and issues that could impact the security posture. The review for this project was performed using manual methods and utilizing the experience of the reviewer. No dynamic testing was performed, only the use of custom-built scripts and tools was used to assist the reviewer during the testing. We discuss our methodology in more detail in the following subsections.

Code Review

Kudelski Security Team reviewed the code within the project utilizing an appropriate IDE. During every review, the team spends considerable time working with the client to determine correct and expected functionality, business logic, and content, to ensure that findings incorporate this business logic into each description and impact. Following this discovery phase, the team works through the following categories:

- authentication (*e.g.* A07:2021, CWE-306)
- authorization and access control (*e.g.* A01:2021, CWE-862)
- auditing and logging (*e.g.* A09:2021)
- injection and tampering (*e.g.* A03:2021, CWE-20)
- configuration issues (*e.g.* A05:2021, CWE-798)
- logic flaws (*e.g.* A04:2021, CWE-190)
- cryptography (*e.g.* A02:2021)

These categories incorporate common weaknesses and vulnerabilities such as the OWASP Top 10 and MITRE Top 25.

Smart Contracts

We reviewed the smart contracts, checking for additional specific issues that can arise such as:

- risk of centralization
- reentrancy
- (non)-adherence to existing standards
- unsafe arithmetic operations

4.4 Reporting

Kudelski Security delivered to the Client a preliminary report in PDF format that contained an executive summary, technical details, and observations about the project.

In the report we not only point out security issues identified but also observations for improvement. The findings are categorized into several buckets, according to their overall severity: **Critical**, **High**, **Medium**, **Low**.

Observations are considered to be **Informational**. Observations can also consist of code review, issues identified during the code review that are not security related, but are general best practices and steps, that can be taken to lower the attack surface of the project.

The technical details are aimed more at developers, describing the issues, the severity ranking and recommendations for mitigation.

4.5 Verify

After the preliminary findings have been delivered, we verify the fixes applied by the Client. After these fixes were verified, we updated the status of the finding in the report.

The output of this phase is the final report with any mitigated findings noted.

5. VULNERABILITY SCORING SYSTEM

Kudelski Security utilizes a custom approach when computing the vulnerability score, based primarily on the **Impact** of the vulnerability and **Likelihood** of an attack.

Each metric is assigned a ranking of either low, medium or high, based on the criteria defined below. The overall severity score is then computed as described in the next section.

Severity

Severity is the overall score of the finding, weakness or vulnerability as computed from Impact and Likelihood. Other factors, such as availability of tools and exploits, number of instances of the vulnerability and ease of exploitation might also be taken into account when computing the final severity score.

IMPACT / LIKELIHOOD	LOW	MEDIUM	HIGH
HIGH	MEDIUM	HIGH	HIGH
MEDIUM	LOW	MEDIUM	HIGH
LOW	LOW	LOW	MEDIUM

Compute overall severity from Impact and Likelihood. The final severity factor might vary depending on a project's specific context and risk factors.

- **Critical** The identified issue may be immediately exploitable, causing a strong and major negative impact system-wide. They should be urgently remediated or mitigated.
- **High** The identified issue may be directly exploitable causing an immediate negative impact on the users, data, and availability of the system for multiple users.
- **Medium** The identified issue is not directly exploitable but combined with other vulnerabilities may allow for exploitation of the system or exploitation may affect singular users. These findings may also increase in severity in the future as techniques evolve.
- **Low** The identified issue is not directly exploitable but raises the attack surface of the system. This may be through leaking information that an attacker can use to increase the accuracy of their attacks.
- **Informational** findings are best practice steps that can be used to harden the application and improve processes. Informational findings are not assigned a severity score and are classified as Informational instead.

Impact

The overall effect of the vulnerability against the system or organization based on the areas of concern or affected components discussed with the client during the scoping of the engagement.

- **High** The vulnerability has a severe effect on the company and systems or has an effect within one of the primary areas of concern noted by the client.
- **Medium** It is reasonable to assume that the vulnerability would have a measurable effect on the company and systems that may cause minor financial or reputational damage.
- **Low** There is little to no effect from the vulnerability being compromised. These vulnerabilities could lead to complex attacks or create footholds used in more severe attacks.

Likelihood

The likelihood of an attacker discovering a vulnerability, exploiting it, and obtaining a foothold varies based on a variety of factors including compensating controls, location of the application, availability of commonly used exploits, difficulty of exploitation and institutional knowledge.

- **High** It is extremely likely that this vulnerability will be discovered and abused.
- **Medium** It is likely that this vulnerability will be discovered and abused by a skilled attacker.
- **Low** It is unlikely that this vulnerability will be discovered or abused when discovered.

6. CONCLUSION

The objective of this code review was to evaluate the overall security of the code base and identify any vulnerabilities that would put the product at risk.

The Kudelski Security Team identified 5 security issues: 1 high risk and 4 low risks. On average, the effort needed to mitigate these risks is estimated as `low`.

In order to mitigate the risks posed by this engagement's findings, the Kudelski Security Team recommends applying the following best practices:

- Add minimum `order size` requirement. E.g. minimum size of Jupiter limit order is $5.
- Create an emergency state and check whether the state is enabled or not before the fill function is called.

The Client acknowledged all these vulnerabilities and observations in the follow-up revision of the codebase.

Kudelski Security remains at your disposal should you have any questions or need further assistance.

Kudelski Security would like to thank 1inch Limited for their trust, help and support over the course of this engagement and is looking forward to cooperating in the future.

7. REFERENCES

- Solana Intent Swap Protocol - 1inch Fusion, Draft 1, February 2025
- Whitepaper: https://github.com/1inch/solana-fusion-protocol/pull/72

DOCUMENT RECIPIENTS

NAME	POSITION	CONTACT INFORMATION
Gleb Alekseev		g.alekseev@1inch.io

KUDELSKI SECURITY CONTACTS

NAME	POSITION	CONTACT INFORMATION
Jean-Sebastien Nahon	Application and Blockchain Security Practice Manager	jean-sebastien.nahon@kudelskisecurity.com
Ana Acero	Project Manager/ Operations Coordinator	ana.acero@kudelskisecurity.com

DOCUMENT HISTORY

VERSION	DATE	STATUS/ COMMENTS
V1.0	2 May 2025	Initial draft
V1.1	22 May 2025	Final proposal