Security Research Labs

Interlay Baseline Security Assurance

Threat model and hacking assessment report

V1.0, 21 Mar 2023

Regina Bíró	regina@srlabs.de
Louis Merlin	louis@srlabs.de
Mostafa Sattari	mostafa@srlabs.de
Bruno Produit	bruno@srlabs.de
Gabriel Arnautu	gabriel@srlabs.de
Haroon Basheer	haroon@srlabs.de
Kevin Valerio	kevin@srlabs.de

Abstract. This work describes the result of the thorough and independent security assurance audit of the Interlay parachain platform performed by Security Research Labs. Security Research Labs is a consulting firm that has been providing specialized audit services in the Polkadot ecosystem since 2019, including for the Substrate and Polkadot projects.

During this study, Interlay provided access to relevant documentation and supported the research team effectively. The code of Interlay was verified to assure that the business logic of the product is resilient to hacking and abuse.

The research team identified several issues ranging from informational to critical severity, many of which concerned the Zenlink logic and incorrect extrinsic weights and benchmarking. Interlay, in cooperation with the auditors, already remediated a subset of the identified issues.

In addition to mitigating the remaining open issues, Security Research Labs recommends increasing security maturity by addressing all the open TODOs and reported vulnerabilities on the benchmarking and weight implementation. Further suggested defensive measures include replacing the Zenlink pallet with a more mature decentralized exchange pallet and hardening the collator selection scheme.

- Confidential, 18 pages -

Content

1 Motivation and scope

Blockchains evolve in a trustless and decentralized environment, which by its own nature could lead to security issues. Ensuring availability and integrity is a priority for Interlay as it bridges the Bitcoin and Polkadot ecosystems and provides decentralized finance using Bitcoin as collateral. As such, a security review of the project should not only highlight the security issues uncovered during the audit process, but also bring additional insights from an attacker's perspective, which the Interlay team can then integrate into their own threat modeling and development process to enhance the security of the product.

Interlay is a blockchain network built on top of Substrate. Like other Substrate-based blockchain networks, the Interlay code is written in Rust, a memory safe programming language. Mainly, Substrate-based chains utilize three technologies: a WebAssembly (WASM) based runtime, decentralized communication via libp2p, and a block production engine.

The Interlay runtime consists of multiple modules compiled into a WASM Binary Large Object (blob) that is stored on-chain. Nodes execute the runtime code either natively or will execute the on-chain WASM blob.

The core business logic of Interlay is a bridge from the Bitcoin blockchain network to the Polkadot and Kusama ecosystem. The chain also aims to include multiple decentralized finance services such as loans and a decentralized exchange.

Security Research Labs collaborated with the Interlay team to create an overview containing the runtime modules in scope and their audit priority. The in-scope components and their assigned priorities are reflected in Table 1. During the audit, Security Research Labs used a threat model [1] to guide efforts on exploring potential security flows and realistic attack scenarios. Additionally, Interlay's online documentation [2] provided the testers with a good runtime module design and implementation overview.

Repository	Priority	Component(s)
InterBTC	High	loans nomination btc-relay PR 790 (vault capacity model) [3] runtime configurations
	Medium	annuity bitcoin clients-info collator-selection currency democracy escrow farming fee issue oracle

		redeem replace reward security staking supply traits tx-pause
Zenlink-DEX-Module	High	zenlink-gauge zenlink-protocol zenlink-stable-amm zenlink-swap-router zenlink-vault

Table 1. In-scope Interlay components with audit priority

2 Methodology

This report details the baseline security assurance results for the Interlay parachain with the aim of creating transparency in four steps, treat modeling, security design coverage checks, implementation baseline check and finally remediation support:

Threat Modeling. The threat model is considered in terms of *hacking incentives*, i.e., the motivations to achieve the goals of breaching the integrity, confidentiality, or availability of Interlay parachain nodes. For each hacking incentive, hacking *scenarios* were postulated, by which these goals could be achieved. The threat model provides guidance for the design, implementation, and security testing of Interlay. Our threat modeling process is outlined in Chapter 3.

Security design coverage check. Next, the Interlay design was reviewed for coverage against relevant hacking scenarios. For each scenario, the following two aspects were investigated:

a. **Coverage**. Is each potential security vulnerability sufficiently covered?

b. **Underlying assumptions**. Which assumptions must hold true for the design to effectively reach the desired security goal?

Implementation baseline check. As a third step, the current Interlay implementation was tested for openings whereby any of the defined hacking scenarios could be executed.

To effectively review the Interlay codebase, we derived our code review strategy based on the threat model that we established as the first step. For each identified threat, hypothetical attacks were developed and mapped to their corresponding threat category, as outlined in Chapter 3.

Prioritizing by risk, the codebase was assessed for present protections against the respective threats and attacks as well as the vulnerabilities that make these attacks possible. For each threat, the auditors:

1. Identified the relevant parts of the codebase, for example the relevant pallets and the runtime configuration.

2. Identified viable strategies for the code review. Manual code audits, fuzz testing, and manual tests were performed where appropriate.

3. Ensured the code did not contain any vulnerabilities that could be used to execute the respective attacks, otherwise, ensured that sufficient protection measures against specific attacks were present.

4. Immediately reported any vulnerability that was discovered to the development team along with suggestions around mitigations.

We carried out a hybrid strategy utilizing a combination of code review and dynamic tests (e.g., fuzz testing) to assess the security of the Interlay codebase.

While fuzz testing and dynamic tests establish a baseline assurance, the focus of this audit was a manual code review of the Interlay codebase to identify logic bugs, design flaws, and best practice deviations. We reviewed the Interlay repository up to the commit *ba7b2344503afdff6c003ebc03737309772d1990*. The approach of the review was to trace the intended functionality of the runtime modules in scope and to assess whether an attacker can bypass/misuse/abuse these components or trigger unexpected behavior on the blockchain due to logic bugs or missing checks. Since the Interlay codebase is entirely open source, it is realistic that a malicious actor would analyze the source code while preparing an attack.

Fuzz testing is a technique to identify issues in code that handles untrusted input, which in Interlay's case is extrinsics in the runtime. (Note that the network part is handled by Substrate, which was not in scope for this review, but is built with a strong emphasis on security and where fuzz testing is also used). Fuzz testing works by taking some valid input for a method under test, applying a semi-random mutation to it, and then invoking the method under test again with this semi-valid input. Through repeating this process, fuzz testing can unearth inputs that would cause a crash or other undefined behavior (e.g., integer overflows) in the method under test. The fuzz testing methods written for this assessment utilized the test runtime Genesis configuration as well as mocked externalities to execute the fuzz test effectively against the extrinsics in scope.

Remediation support. The final step is supporting Interlay with the remediation process of the identified issues. Each finding was documented and published with mitigation recommendations. Once the mitigation solution is implemented, the fix is verified by the auditors to ensure that it mitigates the issue and does not introduce other bugs.

During the audit, findings were shared via a private GitHub repository [4]. We also used a private Discord channel for asynchronous communication and status updates – in addition, weekly jour fixe meetings were held to provide detailed updates and address open questions.

3 Threat modeling and attacks

The goal of the threat model framework is to be able to determine specific areas of risk in Interlay's blockchain system. Familiarity with these risk areas can provide guidance for the design of the implementation stack, the actual implementation of the stack, as well as the security testing. This section introduces how risk is defined and provides an overview of the identified threat scenarios. The *Hacking Value*, categorized into *low*, *medium*, and *high*, considers the incentive of an attacker, as

well as the effort required by an attacker to successfully execute the attack. The hacking value is calculated as:

$$Hacking\ Value = \frac{Incentive}{Effort}$$

While *incentive* describes what an attacker might gain from performing an attack successfully, *effort* estimates the complexity of this same attack. The degrees of incentive and effort are defined as follows:

Incentive:

- Low: Attacks offer the hacker little to no gain from executing the threat.
- Medium: Attacks offer the hacker considerable gains from executing the threat.
- High: Attacks offer the hacker high gains by executing this threat.

Effort:

- Low: Attacks are easy to execute. They require neither elaborate technical knowledge nor considerable amounts of resources.
- Medium: Attacks are difficult to execute. They might require bypassing countermeasures, the use of expensive resources or a considerable amount of technical knowledge.
- High: Attacks are difficult to execute. The attacks might require in-depth technical knowledge, vast amounts of expensive resources, bypassing countermeasures, or any combination of these factors.

Incentive and Effort are divided according to Table 2.

Hacking Value	Low incentive	Medium Incentive	High Incentive
High effort	Low	Medium	Medium
Medium effort	Medium	Medium	High
Low effort	Medium	High	High

Table 2. Hacking value measurement scale.

Hacking scenarios are classified by the risk they pose to the system. The risk level, also categorized into low, medium, and high, considers the hacking value, as well as the damage that could result from successful exploitation. The risk of a threat scenario is calculated by the following formula:

$$Risk = Damage \times Hacking\ Value = \frac{Damage \times Incentive}{Effort}$$

Damage describes the negative impact that a given attack, performed successfully, would have on the victim. The degrees of damage are defined as follows:

Damage:

- Low: Risk scenarios would cause negligible damage to the Interlay network
- Medium: Risk scenarios pose a considerable threat to Interlay's functionality as a network.
- High: Risk scenarios pose an existential threat to Interlay's network functionality.

Damage and Hacking Value are divided according to Table 3.

Risk	Low hacking value	Medium hacking	High hacking
Low damage	Low	Medium	Medium
Medium damage	Medium	Medium	High
High damage	Medium	High	High

Table 3. Risk measurement scale

After applying the framework to the Interlay system, different threat scenarios according to the CIA triad were identified.

The CIA triad describes three security promises that can be violated by a hacking attack, namely confidentiality, integrity, availability.

Confidentiality:

Confidentiality threat scenarios concern sensitive information regarding the blockchain network and its users. Native tokens are units of value that exist on the blockchain - confidentiality threat scenarios include for example attackers abusing information leaks to steal native tokens from nodes participating in the Interlay ecosystem and claiming the assets (represented in the token) for themselves.

Integrity:

Integrity threat scenarios threaten to disrupt the functionality of the entire network by undermining or bypassing the rules that ensure that Interlay transactions/operations are fair and equal for each participant. Undermining Interlay's integrity often comes with a high monetary incentive, like for example, if an attacker can double spend or mint tokens for themselves. Other threat scenarios do not yield an immediate monetary reward, but rather, could threaten to damage Interlay's functionality and, in turn, its reputation. For example, invalidating already executed transactions would violate the core promise that transactions on the blockchain are irreversible.

Availability:

Availability threat scenarios refer to compromising the availability of data stored by the Interlay network as well as the availability of the network itself to process normal transactions. Important threat scenarios regarding availability for blockchain systems include Denial of Service (DoS) attacks on participating nodes, stalling the transaction queue, and spamming.

Table 4 provides a high-level overview of the hacking risks concerning Interlay with identified example threat scenarios and attacks, as well as their respective hacking value and effort. The complete list of threat scenarios identified along with attacks that enable them are described in the threat model deliverable [1]. This list can serve as a starting point to the Interlay developers in order to guide their security outlook for future feature implementations. By thinking in terms of threat scenarios and attacks during code review or feature ideation, many issues can be caught or even avoided altogether.

For Interlay, the auditors attributed the most hacking value to the integrity class of threats. Since the efforts required to exploit this kind of issue is considered lower, we identified threat scenarios to the integrity of Interlay as of the highest risk category. Undermining the integrity of the Interlay chain means making unauthorized modifications to the system. Some of the scenarios can have a direct effect on the financials of the system. This can include market manipulation, gaining tokens for free or as a vault stealing collateral without repercussions.

Security promise	Hacking value	Example threat scenarios	Hacking effort	Example attack ideas
Confidentiality	High	- Compromise a user's private key	High	- Targeted attacks to compromise a user's private key
Integrity	High	- Manipulate oracles - Steal collateral as a vault	Medium	- Exploit logic bug to manipulate oracle data to get financial rewards - Exploit bug to bypass stake requirement
Availability	Medium	- Force vaults to go offline - Compromise market availability	Low	- DoS attack against other vaults - Provide a fake Bitcoin fork to the BTC-relay to use the attacker's own blockchain data

Table 4. Risk overview. The threats for Interlay's blockchain were classified using the CIA security triad model, mapping threats to the areas: (1) Confidentiality, (2) Integrity, and (3) Availability.

4 Findings summary

We identified 9 issues - summarized in Table 5- during our analysis of the runtime modules in scope in the Interlay codebase that enable some of the attacks outlined above. In summary, 1 critical severity, 1 high severity, 1 medium severity, 1 low severity and 4 info severity issues were found.

Please note that in our methodology, critical severity issues refer to high severity issues that could be exploited immediately by an attacker on already deployed infrastructure, including a parachain or a non-incentivized testnet.

Issue	Severity	References	Status
Extrinsics with incorrect weights in DEX pallets	Critical	[5] [6]	Partially mitigated in [7]
Incorrect benchmarking calculations in escrow crate	High	[8]	Mitigated in [9]
Extrinsics with missing storage deposits could clutter the blockchain	Low	[10]	Partially mitigated in [9]
Legitimate democracy proposal's metadata can be attacker-controlled	Low	[11]	Mitigated in [12]
Missing route checking during token swapping	Info	[13]	Mitigated in [14]
Code relies on Enum byte representation instead of Enum variant	Info	[15]	Mitigated in [12]
Collator selection mechanism gives an unfair advantage to entities possessing a disproportionate quantity of tokens	Info	[16]	Chain is not affected as of now
Collateral price can be manipulated to get outsized loan	Info	[17]	Chain is not affected anymore

Table 5 Issue summary

5 Detailed findings

5.1 Extrinsics with incorrect weights in DEX pallets

Attack scenario	**DoS via triggering block production timeouts caused by incorrect extrinsic benchmarking**
Location	zenlink-gauge, zenlink-protocol, zenlink-stable-amm, zenlink-vault
Tracking	https://github.com/interlay/srlabs-audit/issues/2 https://github.com/interlay/srlabs-audit/issues/10
Attack impact	Extrinsics in the DEX pallets have underestimated weights which can be abused to stall the chain.
Severity	Critical
Status	Partially mitigated in [7]

Interlay is using the Zenlink pallets for the DEX (decentralized exchange) functionality. A DEX works by creating liquidity pools to which users contribute with their assets.

The absence of appropriate benchmarking in the Zenlink pallets enables users to execute extrinsics with fees that do not correspond to their execution time. This can be exploited by an attacker to create timeout errors on validator nodes, potentially resulting in the temporary halting of the chain even leading to DoS until a governance intervention from the parachain. The recommended solution is to implement adequate benchmarking for the Zenlink pallets, using a worst-case approach. This can be done by using a *BoundedVec* structure and or making the weight dependent on the length of the parametric route.

The attached issue [5] provides a proof-of-concept in Python that triggers the creation of a block that lasts for multiple seconds with an extrinsic call that has a small weight. The proof-of-concept swaps iBTC for kBTC using a large chain of iBTC/kBTC – kBTC/iBTC swaps (30000 routes).

While this type of vulnerability is already tracked by Interlay's team, the report aims to emphasize the complete risks of incorrect benchmarking.

The issue was reported by Security Research Labs in two steps. In the first issue it was outlined that some weights in the DEX pallets were hardcoded (e.g., "10000") [6], which deviating from Substrate guidelines. In the second issue reported with critical severity, an actual exploit of an underweighted extrinsic was presented [5].

5.2 Incorrect benchmarking calculations in escrow crate

Attack scenario	**Delay block creation using underweighted extrinsics**
Location	escrow
Tracking	https://github.com/interlay/srlabs-audit/issues/4
Attack impact	Delays in block creation and missed validator slots.
Severity	High
Status	Mitigated in [9]

The escrow pallet is designed to allow users to lock their native currency and receive tokens that are vote-escrowed. This voting power is reduced in a linear fashion over time, and the lockup period ends at a predetermined block height.

A malicious actor could execute underweighted extrinsics (such as *create_lock*, *increase_amount*) in the escrow pallet at a low cost. If exploited by using *batch_all,* the impact of the attack will be amplified – this could result in the collators/validators missing their slots and causing delays in block creation. Consequently, important system configuration calls issues by the governance could be delayed, as well as normal user transactions, which could lead to loss of trust in the blockchain network. Please find additional details on exploiting this vulnerability in the respective issue report [8].

To address this issue, the benchmarking functions should be updated to reflect the worst-case computational complexity scenario weight. The benchmarking investigation done by the auditors suggests that the weight of these extrinsics should be increased by at least two orders of magnitude.

5.3 Extrinsics with missing storage deposits could clutter the blockchain

Attack scenario	Clutter up blockchain storage to exclude nodes from block production
Location	fee
Tracking	https://github.com/interlay/srlabs-audit/issues/5
Attack impact	Runtime storage becomes cluttered.
Severity	Low
Status	Partially mitigated in [9]

The set_commission function sets the commission for a given vault and currency pair. A malicious actor could exploit the pallet by repeatedly calling the *set_commission* extrinsic and storing irrelevant data into the blockchain database.

In that extrinsic, the size of an insertion is computed as the size of a commission in the memory plus the size of the key, making a total size of 256 bits/insertion. An average call to the extrinsic costing 9.9498 milli INTR, an attacker with $1000 budget could exhaust the database up to 90.7Mb. Performing such an attack could result in the blockchain becoming cluttered and overburdened, making it harder to operate. At some point, it would become infeasible to keep up with the storage requirements to run a node.

It is recommended to impose a storage fee or deposit on any transaction that writes data to the storage database, to prevent malicious actors from clogging up the blockchain storage. Once the data is removed from the storage database, the deposits should be returned to the caller of the extrinsic. This is especially important for large deposits.

As part of [9], rigorous benchmarks were applied to the affected extrinsics. This helps mitigate the issues. Consequently, we lowered the severity from Medium to Low.

5.4 Legitimate democracy proposal's metadata can be attacker-controlled

Attack scenario	Manipulate opinion by being the first to submit information about a proposal
Location	democracy
Tracking	https://github.com/interlay/srlabs-audit/issues/6
Attack impact	Confusing users and preventing proposal abuses.
Severity	Low
Status	Mitigated in [12]

The democracy pallet contains the core logic linked to community decisions made on a parachain, such as upgrades and configuration modifications through submitting proposals and enacting them via referenda.

This vulnerability allows attackers to manipulate user opinion by submitting a public proposal with a known hash before it is legitimately proposed. This permits an attacker to set a preimage, such as a false forum URL or document, to mislead or confuse users and impede a proposal from achieving a majority.

To prevent this, we suggest linking the preimage to the *ProposalIndex* instead of the hash, allowing each proposer to set preimage for their own proposal regardless of concurrent proposals with the same hash.

This issue has been fixed in [12] by using the upstream fixes of Parity's democracy pallet. We recommend relying on this upstream code directly to get all security updates with every release.

5.5 Missing route checking during token swapping

Attack scenario	**Mislead users through impossible trade routes**
Location	zenlink-swap-router
Tracking	https://github.com/interlay/srlabs-audit/issues/9
Attack impact	Because of non-existent checks, a malicious exchange could offer misleading exchange rates through swap route manipulation.
Severity	Info
Status	Mitigated in [14]

The zenlink-swap-router pallet allows invalid routes when swapping through stable pool. This is because there is no verification that the last value of one route matches the first value of the next route, potentially resulting in unintended trades.

For example, a malicious exchange could craft a very legitimate-looking transaction for a user to sign, which includes trades that are totally out of the scope of the target swap. This could be used to manipulate markets or to offer "better exchange rates" than other exchanges, without the user realizing that they are trading more than one currency to get to the given outcome.

Because this attack does not fit in our threat model, we reported this issue with the *Info-level* severity. To mitigate the issue, we suggest implementing a check that verifies that the routes correctly follow each other.

This issue has been fixed in [14]. The fix verifies that the input of each route matches the output of the previous route in the list, which completely mitigates the issue.

5.6 Code relies on Enum byte representation instead of Enum variant

Attack scenario	**Break storage by abusing code inconsistency**
Location	democracy
Tracking	https://github.com/interlay/srlabs-audit/issues/7
Attack impact	PreImageStatus changes could lead to incorrect behavior.
Severity	Info
Status	Mitigated in [12]

In the democracy pallet the *check_pre_image_is_missing* function code relies on hardcoded magic values instead of transforming the Enum value to an integer. This implementation presents a potential risk, because if the *PreImageStatus* Enum changes in the future, the developer may forget to alter the relevant code. To mitigate this risk, it is suggested to convert the respective Enum values to integers

instead of relying on hardcoded values. By doing so, the code would be able to adjust to changes in the *PreImageStatus* Enum automatically.

This issue has been fixed in [12].

5.7 Collator selection mechanism gives an unfair advantage to entities possessing a disproportionate quantity of tokens

Attack scenario	**Manipulate block collation by acquiring many tokens**
Location	collator-selection
Tracking	https://github.com/interlay/srlabs-audit/issues/8
Attack impact	A whale could abuse the collator selection mechanism.
Severity	Info
Status	Chain is not affected as of now

The mechanism implemented in collator-selection could be exploited by whales – entities possessing a disproportionate quantity of tokens – to fill all the non-invulnerable collator candidate slots, potentially allowing them to control more than 50% of all collators and thus manipulate transaction order, perform frontrunning and temporarily censor transactions in their respective slots for block creation. While the shared security model of the Polkadot ecosystem only requires at least one honest collator for a parachain to prevent complete takeover of the chain, having a large proportion of malicious collators could potentially hinder important business operations and undermine trust in the parachain service.

To avoid this, it is recommended to ensure a large percentage of honest collators when configuring this pallet (the number of configured Invulnerables should be always higher than the number on external non-Invulnerables), and to consider implementing a nomination-based collator selection mechanism with a minimum candidacy bond. Moonbeam's DPoS (Delegated Proof of Stake) nomination scheme for collator selection [18] can be used as a reference.

5.8 Collateral price can be manipulated to get outsized loan

Attack scenario	**Gain outsized rewards by pumping collateral token**
Location	loans
Tracking	https://github.com/interlay/srlabs-audit/issues/3
Attack impact	Draining lending pools by utilizing inflated collateral value.
Severity	Info
Status	Chain is not affected anymore

To open a borrow position, collateral must be provided. Lending positions can be used as a collateral, permitting earning rewards on that position. Kintsugi token (KINT), Interlay's official canary network token, can be used as a collateral for any lending position. However, an attacker could manipulate a thin and illiquid order-book of the KINT token to artificially inflate the price of the collateral and thus gain access to larger loans than they would be entitled to under normal circumstances. Here is an example of such an attack scenario:

1. The adversary buys KINT, takes a lending position and uses it as a collateral,

2. The adversary does a market-buy order, sweeps the selling order-book, and artificially pumps the price of KINT in a centralized exchange with low-liquidity: the value of the collateral rises while the financial cost for the pump is low,

3. The adversary can now borrow an outsized amount of USDT with the collateral acquired in step 1.

As an example, on January 27th, 2023, the volume of the KINT/USDT pair on Gate.io used to represent 65% of the total volume. On this date, it would have taken approximately 8800 USDT of market-buy order to sweep the selling order-book and artificially pump the price from 0.95 USDT to 1.87 USDT, an increase of approximately 97%, allowing an attacker to almost double his collateral.

The issue was addressed by removing the KINT token from the lending market, however the potential risk of similar scenarios should be considered to prevent similar attacks in the future.

Potential mitigation measures include increasing the collateral requirements depending on market liquidity, handling periods of high volatility, increasing liquidity in the KINT pairs on exchanges, and monitoring unusual trading behavior and volume spikes.

6 Evolution suggestions

The overall impression of the auditors was that Interlay as a product is designed and written with security in mind. To ensure that Interlay is secure against unknown or yet undiscovered threats, we recommend considering the evolution suggestions and best practices described in this section.

6.1 Core business logic improvement suggestions

Consider utilizing a more mature DEX Rust library for the decentralized exchange in Interlay. The code implemented by the Zenlink team offers basic DEX functionality. However, many common practices for Substrate pallets are not respected:

- Testing is not extensive.
- Benchmarking is not reflecting the worst-case for many extrinsics.
- Origins are not configurable (Root is the authority for every major decision).
- Code uses unsafe patterns, such as unbounded vectors.

Parity is currently working on a DEX pallet [19] of their own for the ecosystem. Security audit companies will be looking at it in the coming months, after which it should be safe to use. This pallet will not have the downsides listed above.

Secure collator selection mechanism to prevent manipulation by malicious actors. Interlay currently uses a fork of the upstream collator-selection pallet in cumulus, with only Invulnerable collators active on the live chains. The pallet gives the possibility to parachains to allow anyone external to register as a collator, in

exchange for a certain amount of stake. The number of these collators is a configurable parameter exposed to the runtime configuration of the parachain. As described in Section 5.6, in case the proportion of non-invulnerable collators is high and a whale can register all the available collator slots, block production on the parachain could become undemocratic and could be prone to censorship attacks. To prevent this, we strongly advise to always make sure that a single entity cannot control a large proportion of collators. We recommend considering a staking nomination-based collator selection approach.

Ensure high liquidity in all financial functions on Interlay and monitor suspicious behavior. As highlighted in Section **Error! Reference source not found.**, it is of utmost importance that Interlay ensures enough liquidity to all financial services, in decentralized liquidity pools as well as in the different traded pairs on centralized exchanges, to prevent any abuses from malicious entities such as price manipulation. Any usage of illiquid collateral is highly not recommended to avoid any slippage issues during potential liquidations and high volatility periods.

6.2 Address currently open security issues

We recommend addressing already known security issues in a timely manner to prevent attackers from exploiting them – even if an open issue has a limited impact, an attacker might use it as part of their exploitation chain, which may have a more severe impact on Interlay.

Address TODOs and vulnerabilities in the open issue tracker. As discussed during the audit, Interlay has already been aware of certain security issues, like missing/incorrect benchmarking and weights or misconfigured *ExistentialDeposit*, which are tracked in the public repository [20]. We encourage Interlay to address these issues in a timely manner to ensure that these openly tracked vulnerabilities do not serve as hints to attackers. Moreover, we recommend revisiting the current benchmarking implementation to ensure it covers worst case computational complexity scenarios, as showcased in Section 5.2 and implementing storage deposits for all affected pallets.

In the case of keeping Zenlink, ensure that all reported vulnerabilities are fixed. As described in Section 6.1, we recommend using a more mature DEX Rust library than Zenlink. In the case Interlay would prefer to keep using Zenlink, we strongly suggest that all open reported vulnerabilities for this pallet (described in Sections 5.1 and 5.2) are mitigated. Furthermore, we encourage Interlay to conduct an economic audit including the Zenlink implementation.

6.3 Further recommended best practices

Avoid forking pallets. While it may not always be possible to contribute upstream to popular pallets, such as the democracy pallet [21], the loans pallet [22] or the Zenlink DEX pallets [23], forking should be avoided in most cases. Having a fork of a known pallet makes getting upstream fixes a manual process and harder to maintain. Moreover, the adapted fix for the forked pallet may not mitigate the underlying security issue, or it may introduce new vulnerabilities.

Engage in an economic audit. Although SRLabs has some knowledge of economic attacks, our primary goal during engagements is to find logic vulnerabilities through code assurance. Therefore, an economic audit for the loans, vault, and DEX pallets,

and their configuration is recommended to ensure the safety of Interlay's platform and users.

Regular code review and continuous fuzz testing. Regular code reviews are recommended to avoid introducing new logic or arithmetic bugs, while continuous fuzz testing can identify potential vulnerabilities early in the development process. Ideally, Interlay should continuously fuzz their code on each commit made to the codebase. The Polkadot codebase provides a good example of multiple fuzzing harnesses based on *honggfuzz* [24].

Regular updates. New releases of Substrate may contain fixes for critical security issues. Since Interlay is a product that heavily relies on Substrate, updating to the latest version as soon as possible whenever a new release is available is recommended. Although Interlay has been diligent in upgrading Substrate versions (the current development of interBTC is only two minor Substrate versions behind), it is essential to keep up this effort.

7 Bibliography

[1] [Online]. Avail
https://securityresearchlabs.sharepoint.com/:x:/s/Interlay/ERwjAJn7UGdEi4fEO2nC6v0BaX7Oa4jUqxvbyFppPUc

[2] [Online]. Available: https://spec.interlay.io/.

[3] [Online]. Available: https://github.com/interlay/interbtc/pull/790.

[4] [Online]. Available: https://github.com/interlay/srlabs-audit.

[5] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/10.

[6] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/2.

[7] [Online]. Available: https://github.com/interlay/interbtc/pull/966.

[8] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/4.

[9] [Online]. Available: https://github.com/interlay/interbtc/pull/985.

[10] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/5.

[11] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/6.

[12] [Online]. Available: https://github.com/interlay/interbtc/pull/949.

[13] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/9.

[14] [Online]. Available: https://github.com/interlay/interbtc/pull/962.

[15] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/7.

[16] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/8.

[17] [Online]. Available: https://github.com/interlay/srlabs-audit/issues/3.

[18] [Online]. Available: https://github.com/PureStake/moonbeam/tree/master/pallets/parachain-staking.

[19] [Online]. Available: https://github.com/paritytech/substrate/pull/12984.

[20] [Online]. Available: https://github.com/interlay/interbtc/issues?page=1&q=is%3Aissue+is%3Aopen.

[21] [Online]. Available: https://github.com/paritytech/substrate/tree/master/frame/democracy.

[22] [Online]. Available: https://github.com/parallel-finance/parallel/tree/master/pallets/loans.

[23] [Online]. Available: https://github.com/zenlinkpro/Zenlink-DEX-Module.

[24] [Online]. Available: https://github.com/paritytech/polkadot/tree/master/xcm/xcm-simulator/fuzzer.

[25] [Online]. Available: https://github.com/interlay/interbtc/pulls?q=weights.