Security
Research
Labs

Frequency Baseline Security Assurance Report

Threat model and hacking assessment report

v1.0, October 7, 2024

Prepared for:
Frequency

- Confidential, 20 pages -

Content

Disclaimer

This report describes the findings and core conclusions derived from the audit carried out by Security Research Labs within the agreed-on timeframe and scope as detailed in Chapter 3.

Please note that this report does not guarantee that all existing security vulnerabilities were discovered in the codebase exhaustively and that following all evolution suggestions described in Chapter 6 may not ensure all future code to be bug free.

Version:	v1.0	
Prepared For:	Frequency	
Date:	October 7, 2024	
Prepared By:	Gabriel Arnautu	gabriel@srlabs.de
	Marc Heuse	marc@srlabs.de
	Nils Ollrogge	nils@srlabs.de

Timeline

Table 1 shows the timeline of the Frequency source code security audit in 2024.

Date	Event
August 1, 2024	Kick-off security audit
August 19, 2024	Shared draft threat model
October 7, 2024	Shared audit report

Table 1: Audit timeline

Integrity Notice

1 Executive summary

1.1 Engagement overview

This work describes the result of the security assurance audit for the Frequency parachain performed by Security Research Labs, covering the period from 1st of August to 27th of September 2024. Security Research Labs is a consulting firm that has been providing specialized blockchain audit services since 2019.

During this study, the Frequency development team provided access to relevant documentation and supported the research team effectively. The protocol design, concept documentation and relevant available source code of Frequency was verified to assure that the Frequency protocol is resilient to hacking and abuse.

Security Research Labs conducted a comprehensive security assurance audit of the Frequency blockchain’s code, reviewing critical components. This audit focused on assessing Frequency‘s codebase for resilience against hacking and abuse scenarios. The testing approach combined static and dynamic analysis techniques, leveraging both automated tools and manual inspection. Security Research Labs prioritized reviewing critical functionalities and conducting thorough security tests to ensure the robustness of Frequency’s platform.

This report focuses on our assessment results concerning the Frequency runtime.

1.2 Observations and Risk

The research team identified several issues ranging from low to informational level severity. The Frequency team, in cooperation with the auditors, already remediated most of the identified issues.

1.3 Recommendations

Security Research Labs recommends integrating more comprehensive security tests. Additionally, conducting thorough threat modeling, further secure development best-practices and economic audits will identify potential risks and ensure the economic integrity of the system.

2 Evolution suggestions

The overall impression of the auditors is that Frequency follows strong security practices and has made efforts to build a secure codebase. To further enhance Frequency's defenses against potential unknown or emerging threats, we recommend considering the evolution suggestions and best practices described in this section.

2.1 Secure development improvement suggestions

We recommend to further strengthen the security of the blockchain by implementing the following recommendations:

Perform threat modeling. Performing threat modeling for all new features and major updates before coding is crucial for better code security. This practice allows developers to identify potential security threats and vulnerabilities early in the design phase, enabling them to implement appropriate mitigations from the outset. Including the threat model in the pull request description ensures that the entire team is aware of the identified risks and the measures taken to address them, promoting a proactive security culture and enhancing the overall robustness of the codebase. Additionally, it helps the audit team to identify gaps in the threat model and focus their assessment.

Use static analysis. Using static analysis tools to detect security flaws in the codebase is essential for improving code security. These tools, such as Dylint [1] and Semgrep [2] for the Rust ecosystem, analyze the code without executing it, identifying vulnerabilities, coding errors, and compliance issues early in the development process. This proactive approach helps developers address potential security issues before they reach production, ensuring a more secure and reliable codebase.

Perform dynamic analysis. Developing fuzzing harnesses for critical components is essential for identifying security vulnerabilities and business logic issues. By employing invariants, these fuzzing tests can effectively uncover subtle flaws that might otherwise go unnoticed. The Polkadot codebase exemplifies this approach, utilizing multiple fuzzing harnesses based on the substrate-runtime-fuzzer, demonstrating how comprehensive and targeted fuzz testing can significantly enhance the security and reliability of complex systems; see the *substrate-runtime-fuzzer* [3].

Launch a bounty program. Establishing a bounty program to encourage the external discovery and reporting of security vulnerabilities is crucial for enhancing code security. By offering monetary incentives, such programs motivate individuals to responsibly report vulnerabilities to Frequency rather than exploiting them. This approach not only broadens the pool of people actively searching for security flaws but also fosters a collaborative security environment, helping to identify and address issues more quickly and effectively.

2.2 Further recommended best practices

Regular updates. New releases of Polkadot-SDK may contain fixes for critical security issues. Since Frequency is a product that heavily relies on Polkadot-SDK, updating to the latest version as soon as possible whenever a new release is available is recommended.

3 Motivation and scope

Blockchains evolve in a trustless and decentralized environment, which by their very nature can lead to security issues. Ensuring confidentiality and integrity is a priority for Frequency as it aims to provide a platform for scalable, data-focused messaging as a backbone for new data distribution protocols, empowering users with greater transparency, authenticity, and control over their privacy and data. As such, a security review of the project should not only highlight any security issues uncovered during the audit, but also bring additional insights from an attacker's perspective, which the Frequency team can then integrate into their threat modeling and development process to enhance the security of the platform.

The core business logic of Frequency is to provide scalable, data-focused messaging as a backbone for new data distribution protocols such as the Decentralized Social Networking Protocol (DSNP). By introducing a different class of blockchain transactions that are not financial in nature, Frequency distinguishes between financial transactions and data-focused transactions. Data-focused transactions have reduced requirements—like the need to defend against double-spend attacks—while retaining blockchain guarantees such as authenticity and data validation.

Frequency emphasizes broadcast messaging to allow information to be discovered by anyone, shifting control of data discovery from centralized authorities, as seen in traditional systems, to the users themselves. This empowers users to choose their preferred tools and consume only the data they want.

Extending its belief in user ownership to the relationship between users and decentralized applications (dApps), Frequency allows users to trust dApps in limited ways through user delegation. This delegation functionality further enables the option of coinless users, enabling dApps to cover blockchain costs on behalf of users and construct the economic structures they believe are best for their user base.

To address economic challenges like transaction cost volatility, Frequency introduces a staking system called Capacity for sending messages. This system aims to provide better cost predictability for businesses trying to build products on a blockchain.

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

Frequency's 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.

Security Research Labs collaborated with the Frequency 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 . During the audit, Security Research Labs used a threat model to guide efforts on exploring potential security flaws and realistic attack scenarios. Additionally, Frequency's online documentations and specification provided the auditors with a good runtime module design and implementation overview.

Repository	Priority	Component(s)	Reference
Frequency	High	pallet-capacity pallet-frequency-tx-payment pallet-handles	

🟧	pallet-messages
	pallet-msa
	pallet-schemas
	pallet-stateful-storage
	pallet-time-release

Table 2: In-scope Frequency components with audit priority

4 Methodology

This report details the continuous security assurance results for the Frequency network with the aim of creating transparency in four steps: threat modeling, security design coverage checks, implementation baseline check and finally remediation support. We applied the following four steps methodology when performing feature reviews.

4.1 Threat modeling and attacks

The goal of the threat model framework is to be able to determine specific areas of risk in Frequency network. 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 .

Hacking Value	Low incentive	Medium Incentive	High Incentive

High effort	Low	Medium	Medium
Medium effort	Medium	Medium	High
Low effort	Medium	High	High

Table 3: 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 Frequency network.
- Medium: Risk scenarios pose a considerable threat to Frequency‘s functionality as a network.
- High: Risk scenarios pose an existential threat to Frequency network functionality.

Damage and Hacking Value are divided according to Table .

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

Table 4: Risk measurement scale

After applying the framework to the Frequency 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 Frequency ecosystem and claiming the assets for themselves. As a social network protocol, Frequency also needs to ensure user privacy; an attacker who compromises this could damage the reputation of victims by exposing sensitive information or impersonating them.

Integrity:

Integrity threat scenarios threaten to disrupt the functionality of the entire network by undermining or bypassing the rules that ensure that Frequency transactions/operations are fair and equal for each participant. Undermining Frequency's integrity often comes with a high monetary incentive, such as when an attacker can double spend or mint tokens for themselves. Other threat scenarios may not offer immediate monetary rewards but could damage Frequency's functionality and, consequently, its reputation.

Availability:

Availability threat scenarios refer to compromising the availability of data stored by the Frequency 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 provides a high-level overview of the hacking risks concerning Frequency 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. This list can serve as a starting point for the Frequency developers 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.

Security promise	Hacking value	Example threat scenarios	Hacking effort	Example attack ideas
Confidentiality	High	- Compromise a user's private key - Impersonating a user	High	- Create similar service to Frequency Access which manages user's keys - Use a similar user handle
Integrity	High	- Affect economical model of the network - Manipulate the blockchain's history	High	- Redirect Capacity to another provider - Exploit rounding issues
Availability	High	- Spam the blockchain with bogus transactions - Make the network harder to operate	Medium	- Cheaply bloat blockchain storage - Exhaust Capacity tokens of a provider

Table 5: Risk overview. The threats for Frequency' blockchain were classified using the CIA security triad model, mapping threats to the areas: (1) Confidentiality, (2) Integrity, and (3) Availability.

4.2 Security design coverage check.

Next, the Frequency 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?

4.3 Implementation check

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

To effectively review the Frequency 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 4.1.

Prioritizing by risk, the code 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, static test is and dynamic tests (e.g., fuzz testing) to assess the security of the Frequency codebase.

While static testing, fuzz testing and dynamic tests establish a baseline assurance, the focus of this audit was a manual code review of the Frequency codebase to identify logic bugs, design flaws, and best practice deviations. We reviewed the audit-2024 branch of the Frequency repository which contains the Frequency runtime implementation up to commit *9ef5818* from the *7th of August 2024.* 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 Frequency 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 Frequency‘s case are extrinsics in the runtime. (Note that the network layer is handled by Polkadot-SDK, 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.

4.4 Remediation support

The final step is supporting Frequency 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 their GitHub repository [4]. We also used a private slack channel for asynchronous communication and status updates – in addition, bi-weekly jour fixe meetings were held to provide detailed updates and address open questions.

5 Dynamic analysis assessment

During the audit, we utilized fuzz testing to identify bugs in the runtime of Frequency. By applying fuzz testing, we are aiming to uncover issues that may not have been detected through manual code review. The fuzzing campaign was continuously updated as our understanding of the Frequency runtime evolved. Ultimately, the campaign did not reveal any issues, providing assurance that common programming errors, such as unsafe math operations, were absent from the code.

Fuzz harness creation: We chose our in-house developed and opensource tool Ziggy [5] as our fuzzing orchestration tool. As for the harness, we used a customized harness for the runtime which is comparable to our substrate runtime fuzzing harness that is also available on GitHub [3].

Coverage analysis and optimization: Although modern fuzzers can achieve good coverage by utilizing various techniques, we manually generated some seeds to target specific functionalities that were not covered by the fuzzer after a certain period. This approach assisted the fuzzer and optimized the overall coverage, ensuring more comprehensive testing.

In our coverage analysis below, we only measure the Frequency codebase coverage that the harness is targeting.

5.1 Runtime fuzzer

Component	Code path	Coverage achieved
Frequency Runtime	/runtime/frequency	68.61%

The coverage for the Frequency runtime fuzzer stands at 68.61%, with most lines not covered being either inherent or only root callable extrinsics, for example `set_epoch_length`. There remains room for improvement, particularly in testing passkey related extrinsics, which require additional configuration and setup adjustments in the fuzzing harness.

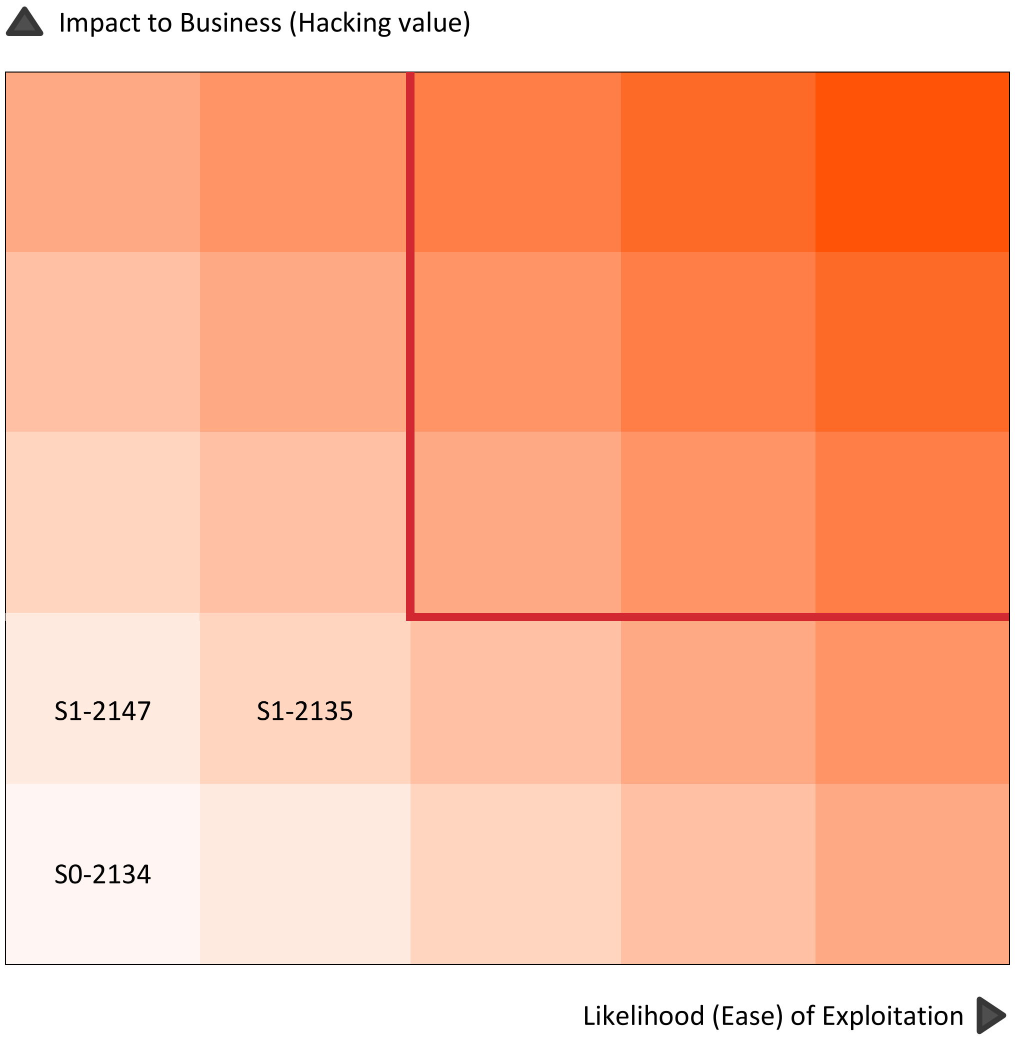
6 Findings summary

We identified 3 issues during our analysis of the runtime modules in scope in the Frequency codebase that enabled some of the attacks outlined below. In summary, 2 low severity and 1 info level issues were found. An overview of all findings can be found in Table .

High	0	
Medium	0	
Low	2	▬ ▬
Informational	1	▬
Total Issues	**3**	

6.1 Risk profile

The chart below summarizes vulnerabilities according to business impact and likelihood of exploitation, increasing to the top right.



6.2 Issue summary

ID	Issue	Severity	Status
S1-2135 [6]	Minimum staking amount not enforced	Low	Mitigated [7]
S1-2147 [8]	Possible user handle confusion	Low	Mitigated [9]
S0-2134 [10]	Possible interoperability issue in passkey signature verification	Info	Mitigated [11]

Table 6: Findings overview

7 Detailed findings

7.1 Minimum staking amount not enforced

Attack scenario	**Minimum staking amount is not enforced if a stake already exists for a different provider**
Component	Frequency-capacity-pallet
Tracking	https://github.com/frequency-chain/frequency/issues/2135
Attack impact	Potential storage bloating
Severity	Low
Status	Mitigated [7]

Context
Providers in the Frequency blockchain are special accounts owned by applications and services. To express or withdraw support for these providers, Frequency allows users to stake FRQCY tokens to them. In return, the system rewards users with tokens and providers with capacity. This process is called provider boosting.

Issue description
The `ensure_can_stake` function is supposed to verify that that the amount the account is staking to a provider exceeds the minimum required staking amount. To do this, the function checks whether the sum of current active staking amount and the new staking amount is bigger than `MinimumStakingAmount`. Since the current active staking amount is accounted for in the check, this implies that a user must only stake the `MinimumStakingAmount` when they initially submit a stake for a provider. All subsequent staking requests will be accepted, even with a staking amount of 1.

Risk
This allows an attacker to stake the minimum required amount only once to a provider and then repeatedly stake just 1 token in subsequent staking request. As a result, the attacker underpays for their storage usage, potentially leading to storage bloating and slowing down the network in the long run.

Mitigation
To mitigate this issue, the `ensure_can_stake` function should verify that the new staking amount exceeds the `MinimumStakingAmount`, independent of the current active staking amount. This ensures that every staking request meets the minimum requirement and prevents misuse of the system.

7.2 Possible user handle confusion

Attack scenario	**A user chooses a username very similar to a reserved one by exploiting Unicode characters.**
Component	Frequency-handles-pallet
Tracking	https://github.com/frequency-chain/frequency/issues/2147
Attack impact	Possible user handle confusion
Severity	Low
Status	Mitigated [9]

Context
The Frequency blockchain allows users to register handles—unique and user-friendly names—to make it easier to identify and interact with their accounts. Each handle is uniquely mapped to a Message Source Account (MSA) ID. To ensure the uniqueness of handles, a numeric suffix is appended to the chosen handle, separated by a delimiter [.].

Issue description
When a user tries to claim a handle, checks are performed to ensure its validity and prevent misuse. The `is_reserved_handle` check compares the requested handle against a list of reserved words, such as “admin” or “moderator”, to prevent users from claiming handles that might carry special meaning. However, this check does not consider the canonical representation of the handle. As a result, users can exploit Unicode characters that resemble Latin alphabet letters to create handles that appear similar reserved words.
On the other hand, the functionality in Frequency that generates numeric suffixes for handles does account for the canonical representation. Consequently, there won’t exist two handles that differ only by similar-looking Unicode characters. This makes the issue low in severity.

Risk
A user could claim a handle with a prefix that closely resemble a reserved handle, potentially misleading others into believing the user holds a special status.

Mitigation
We recommend adjusting the `is_reserved`_handle check to compare against the canonicalized representation of reserved handles. This will ensure that users cannot bypass restrictions by using visually similar Unicode characters.

7.3 Possible interoperability issue in passkey signature verification

Attack scenario	**An application submits signatures without <Byte> wrapping, causing verification failures.**
Component	Frequency-pallet-passkey
Tracking	https://github.com/frequency-chain/frequency/issues/2134
Attack impact	Potential interoperability issues with third-party applications.
Severity	Info
Status	Mitigated [11]

Context
As a convention [12], the polkadot-js-extension always wraps raw data in <Byte> tags to prevent malicious apps from tricking users into signing unintended transactions. The Frequency blockchain ecosystem follows the same approach in both the web application for passkey registration and the runtime code that verifies passkey signatures.

Issue description
Since any entity can develop infrastructure for passkey registration and transaction submission, there is a risk that some implementations may fail to wrap the signature data in <Byte> tags. This poses an issue because the `check_account_signature` function expects binary data to be wrapped in these tags. If the signature data does not follow the expected format, the passkey signature verification will fail, resulting in transaction errors.

Risk
This issue could cause interoperability problems with third-party applications that attempt to interact with the Frequency blockchain but do not adhere to the <Byte> wrapping convention.

Mitigation
To mitigate this, the `check_account_signature` function should attempt to verify signatures using both raw data and raw data wrapped in <Byte> tags, like how it is done in the Substrate NFTs pallet [13]. This would ensure compatibility across different implementations and reduce the risk of signature verification failures.

8 Bibliography

[1] [Online]. Available: https://github.com/trailofbits/dylint.

[2] [Online]. Available: https://github.com/semgrep/semgrep.

[3] [Online]. Available: https://github.com/srlabs/substrate-runtime-fuzzer.

[4] [Online]. Available: https://github.com/frequency-chain/frequency.

[5] [Online]. Available: https://github.com/srlabs/ziggy.

[6] [Online]. Available: https://github.com/frequency-chain/frequency/issues/2135.

[7] [Online]. Available: https://github.com/frequency-chain/frequency/pull/2136.

[8] [Online]. Available: https://github.com/frequency-chain/frequency/issues/2147.

[9] [Online]. Available: https://github.com/frequency-chain/frequency/pull/2161.

[10] [Online]. Available: https://github.com/frequency-chain/frequency/issues/2134.

[11] [Online]. Available: https://github.com/frequency-chain/frequency/pull/2169.

[12] [Online]. Available: https://github.com/polkadot-js/extension/pull/743.

[13] [Online]. Available: https://github.com/paritytech/polkadot-sdk/blob/ebcbca3ff606b22b5eb81bcbfaa9309752d64dde/substrate/frame/nfts/src/common_functions.rs#L41.

Appendix A: Technical services

Security Research Labs delivers extensive technical expertise to meet your security needs. Our comprehensive services include software and hardware evaluation, penetration testing, red team testing, incident response, and reverse engineering. We aim to equip your organization with the security knowledge essential for achieving your objectives.

SOFTWARE EVALUATION We provide assessments of application, system, and mobile code, drawing on our employees' decades of experience in developing and securing a wide variety of applications. Our work includes design and architecture reviews, data flow and threat modelling, and code analysis with targeted fuzzing to find exploitable issues.

BLOCKCHAIN SECURITY ASSESSMENTS We offer specialized security assessments for blockchain technologies, focusing on the unique challenges posed by decentralized systems. Our services include smart contract audits, consensus mechanism evaluations, and vulnerability assessments specific to blockchain infrastructure. Leveraging our deep understanding of blockchain technology, we ensure your decentralized applications and networks are secure and robust.

POLKADOT ECOSYSTEM SECURITY We provide comprehensive security services tailored to the Polkadot ecosystem, including parachains, relay chains, and cross-chain communication protocols. Our expertise covers runtime misconfiguration detection, benchmarking validation, cryptographic implementation reviews, and XCM exploitation prevention. Our goal is to help you maintain a secure and resilient Polkadot environment, safeguarding your network against potential threats.

TELCO SECURITY We deliver specialized security assessments for telecommunications networks, addressing the unique challenges of securing large-scale and critical communication infrastructures. Our services encompass vulnerability assessments, secure network architecture reviews, and protocol analysis. With a deep understanding of telco environments, we ensure robust protection against cyber threats, helping maintain the integrity and availability of your telecommunications services.

DEVICE TESTING Our comprehensive device testing services cover a wide range of hardware, from IoT devices and embedded systems to consumer electronics and industrial controls. We perform rigorous security evaluations, including firmware analysis, penetration testing, and hardware-level assessments, to identify vulnerabilities and ensure your devices meet the highest security standards. Our goal is to safeguard your hardware against potential attacks and operational failures.

CODE AUDITING We provide in-depth code auditing services to identify and mitigate security vulnerabilities within your software. Our approach includes thorough manual reviews, automated static analysis, and targeted fuzzing to uncover critical issues such as logic flaws, insecure coding practices, and exploitable vulnerabilities. By leveraging our expertise in secure software development, we help you enhance the security and reliability of your codebase, ensuring robust protection against potential threats.

PENETRATION & RED TEAM TESTING We perform high-end penetration tests that mimic the work of sophisticated attackers. We follow a formal penetration testing methodology that emphasizes repeatable, actionable results that give your team a sense of the overall security posture of your organization.

SOURCE CODE-ASSISTED SECURITY EVALUATIONS We conduct security evaluations and penetration tests based on our code-assisted methodology, allowing us to find deeper vulnerabilities, logic flaws, and fuzzing targets than a black-box test would reveal. This gives your team a stronger assurance that the significant security-impacting flaws have been found and corrected.

SECURITY DEVELOPMENT LIFECYCLE CONSULTING We guide organizations through the Security Development Lifecycle to integrate security at every phase of software development. Our services include secure coding training, threat modelling, security design reviews, and automated security testing implementation. By embedding security practices into your development processes, we help you proactively identify and mitigate vulnerabilities, ensuring robust and secure software delivery from inception to deployment.

REVERSE ENGINEERING We assist clients with reverse engineering efforts not associated with malware or incident response. We also provide expertise in investigations and litigation by acting as experts in cases of suspected intellectual property theft.

HARDWARE EVALUATION We evaluate new hardware devices ranging from novel microprocessor designs, to embedded systems, to mobile devices, to consumer-facing end products, to core networking equipment that powers Internet backbones.

VULNERABILITY PRIORITIZATION We streamline vulnerability information processing by consolidating data from compliance checks, audit findings, penetration tests, and red team insights. Our prioritization and automation strategies ensure that the most critical vulnerabilities are addressed promptly, enhancing your organization's security posture. By systematically categorizing and prioritizing risks, we help you focus on the most impactful threats, ensuring efficient and effective remediation efforts.

SECURITY MATURITY REVIEW We conduct comprehensive security maturity reviews to evaluate your organization's current security practices and identify areas for improvement. Our assessments cover a wide range of criteria, including policy development, risk management, incident response, and security awareness. By benchmarking against industry standards and best practices, we provide actionable insights and recommendations to enhance your overall security posture and guide your organization toward achieving higher levels of security maturity.

SECURITY TEAM INCUBATION We provide comprehensive support for building security teams for new, large-scale IT ventures. From Day 1, our ramp-up program offers essential security advisory and assurance, helping you establish a robust security foundation. With our proven track record in securing billion-dollar investments and launching secure telco networks globally, we ensure your new enterprise is protected against cyber threats from the start.

HACKING INCIDENT SUPPORT We offer immediate and comprehensive support in the event of a hacking incident, providing expert analysis, containment, and remediation. Our services include detailed forensics, malware analysis, and root cause determination, along with actionable recommendations to prevent future incidents. With our rapid response and deep expertise, we help you mitigate damage, recover swiftly, and strengthen your defences against potential threats.

Appendix B: Risk and advisory services

Security Research Labs enhances an organization's security and risk management capabilities. We offer a practical approach to information security that aligns with our clients' tolerance for security processes and programs. Our team of industry-leading experts brings a diverse range of security and risk management skills, providing clients with deep technical expertise, strategic security leadership, and effective incident response capabilities whenever needed.