Marmot Protocol Review Summary Report

White Noise

Scope

White Noise has requested that Least Authority perform a review of the Marmot Protocol and conduct security audits of both MDK and White Noise in three phases. Marmot combines the MLS (Messaging Layer Service) Protocol with Nostr's decentralized network to provide private group messaging without relying on centralized servers or legacy identity systems. MDK is the Marmot Development Kit. White Noise is the Rust backend of their Flutter application, and the `whitenoise` crate uses the MDK library to implement all components required for a messenger application. This review focuses on the Marmot Protocol.

During this review, we were unable to gain an in-depth understanding of the protocols that Marmot relies on. This only allows us to provide generic best-practice recommendations, which might not fully take into account the specific characteristics and caveats of the subprotocols and subsystems. To increase confidence, we recommend conducting a thorough review of the entire protocol stack.

The following code repositories are considered in scope for the review:

- marmot:
  https://github.com/parres-hq/marmot
    - MIP-05 was not included in this review as it was not complete.

Documentation & Resources

The following documentation and resources are available to the review team of Least Authority:

- Website:
  https://www.whitenoise.chat
- MLS:
  https://datatracker.ietf.org/doc/rfc9420/
- MLS architecture:
  https://www.rfc-editor.org/rfc/rfc9750.html
- Nostr:
  https://github.com/nostr-protocol/nostr
- NIPs:
  https://github.com/nostr-protocol/nips

Areas of Concern

The following are areas of concern that will be investigated during the audit, along with any similar potential issues:

- Coverage and completeness: a general review for comprehensiveness;

- Logical and realistic technical details: a review of basic technical details and the security approach intended to support a rational implementation;
- Potential security issues: an assessment of the threat model and system design with a focus on data confidentiality and system integrity;
- Anything else as identified during the initial analysis phase.

Recommendations

We reviewed the Marmot Protocol, which integrates the Messaging Layer Security (MLS) Protocol to enable private chats and group chats within the decentralized Nostr network.
The positives we identified include the security guarantees provided by MLS, namely post-compromise security, forward secrecy, and non-repudiation, all of which, if correctly implemented, should continue to hold within Marmot.
Nostr provides a decentralized network architecture for clients and relays and is specified through more than 100 NIPs. It supports payload encryption through NIP-44 and introduces seal (signed events) and gift wrap (signed and addressed events) mechanisms through NIP-59. Although earlier NIPs, such as NIP-17, proposed concepts for private direct messaging, NIP-EE represented the first incorporation of MLS over Nostr. Marmot is the successor to NIP-EE.

After reviewing White Noise's Marmot Protocol, our recommendations are as follows:

Recommendation 1: Strengthen Implementation and Usage of the Extractor Key

Due to the prominent usage of the extractor key for Nostr NIP-44 encryption, we recommend placing a special focus on the implementation and usage of the MLS extractor key. While the purpose of the MLS extractor key is specifically for use with outside protocols, it should not be entangled with other keys or reused, since this can lead to the loss of confidentiality in encrypted messages.

Recommendation 2: Produce a High-Level Diagram for Marmot-Specific Events and Packages

Due to the combination of Nostr's decentralized network (with its various NIP possibilities) and the MLS-specific messages and architecture, we recommend producing a diagram illustrating Marmot events, packages, and timelines. The diagram should also include the message structure and the cryptographic protections applied to each field, thereby increasing the protocol's auditability and reducing the risk of missed attacks or connections.

Recommendation 3: Define Dependency Management Requirements

While the Marmot team has indicated that they will not implement the underlying protocols themselves (for example, MLS or Nostr), Marmot is a publicly available specification, and other implementers may choose to use different dependencies, modify existing ones or develop their own implementations.

We recommend that the Marmot specification:

- Define mandatory requirements for library dependencies and protocols;
- Optionally recommend best libraries to other implementers; and
- Advise implementers to periodically monitor their dependencies for newly discovered vulnerabilities.

We recommend applying these practices to all security-critical dependencies.

Recommendation 4: Enforce Key Compartmentalization

We recommend placing strong emphasis on key compartmentalization for White Noise. As the MLS RFC notes, the separation of keys is critical for the security guarantees, such as post-compromise

security, to hold in the event of a compromise. Given the decentralized nature of Nostr, this practice is highly recommended.

Recommendation 5: Define an Explicit and Detailed Threat Model

During our review of the Marmot Protocol and all MIPs, we did not identify an existing threat model. An explicit threat model is important for three primary reasons: it clearly communicates security assurances to users and integrators, supports auditability, and clarifies potential risks along with their mitigations.

We recommend specifying a methodically produced and comprehensive threat model. The threat model should explicitly define the types of attackers it considers. It should also define what the target security and privacy guarantees, providing clear metrics for reasoning about how specific threats can impact Marmot. Additionally, it will help provide advisories to users on how they can mitigate some of the residual risks that Marmot carries.

Below, we provide an example of a high-level threat modelling process for illustration purposes. Since we currently do not have a full understanding of the details of the underlying protocols, specific attack examples provided might not be valid, but they help demonstrate the intended reasoning process.

A non-exhaustive list of threats and their assumed capabilities includes:

- a network observer
    - can capture network packets between clients, relays, and other subsystems
- a malicious relay operator
    - has full control and visibility of one or more relays
- a malicious group participant
    - has access to all group events
- a malicious admin
    - can modify group members and settings
    - can create new groups and invite users
- combinations of the above

As the analysis continues, additional and more specific capabilities will emerge. For example, a network observer could capture Nostr metadata, including Nostr public keys and group tags, since these are sent in plaintext through Nostr events. The next step will be to identify the potential impact and compare it with the target guarantees. In this example, such exposure could enable monitoring of Marmot users and their group affiliations or activities (such as the amount and frequency of messages, timing, or time zone), which reduces privacy for Marmot users.

Another example is that an honest-but-curious relay operator can correlate different ephemeral identities of the same user with their IP address. If this is the case, the system should clearly inform users so they can decide whether to use methods such as frequent IP switching to mitigate the risk.

A last example illustrating why a malicious administrator could be a consideration is that, depending on the precise mechanics of the invitation process, an administrator could create groups and invite specific users to correlate activity metadata. This could enable inference about those users' behavior outside the targeted group. Therefore, it is important to analyze the underlying protocol details governing group invitations and acceptance procedures, as well as their potential overpermissiveness. The same reasoning applies to a malicious nonadministrative participant within a group.

Recommendation 6: Improve Handling of Image Keys for Encryption and Authentication

The Marmot Group Extension, defined in MIP-01, is used to store Marmot-specific group data and includes a group image managed through the Blossom protocol. The image is encrypted using an `image_key` and authenticated through a Nostr key pair. The current Marmot specification recommends setting the `image_key` to the raw output of a cryptographically secure pseudorandom number generator (CSPRNG) rather than processing it through an HKDF. It also states that the Nostr key pair and the `image_key` should remain cryptographically independent. However, the current approach, which derives the Nostr key pair from the `image_key` using an HKDF, creates an undesirable dependency between the two and violates this assumption.

We recommend revising the specification to derive the `image_key` using an HKDF rather than setting it directly to the raw output of a CSPRNG. To maintain true cryptographic independence, the Nostr key pair used for authentication should instead be derived from a distinct seed, independently generated by a CSPRNG.

Recommendation 7: Explicitly Specify Requirements

Marmot provides a method for combining two protocols, MLS and Nostr, for secure messaging purposes. Each of these protocols has its own set of requirements, which are further extended by Marmot's additional requirements for participants, specifically clients and relay nodes. These requirements should be explicitly documented in the Marmot specification. In addition, the Authentication Service (AS), which plays a crucial role in MLS, is not described in the Marmot specification. Given the distributed nature of Marmot and the essential role of the AS in MLS, clearly defining the requirements for this component is crucial to preserving the security assumptions on which MLS relies within Marmot implementations.

Recommendation 8: Specify Administrative Functionality

The Marmot protocol expands the functionality of the MLS protocol by introducing an `Admin` role responsible for approving group changes proposed by other members (including the `Admin`). However, the procedures governing the attribution and management of the `Admin` role throughout a group's lifecycle are insufficiently specified. We recommend expanding the current specification to define how the `Admin` role is attributed, whether it can be modified or revoked, and who has the authority to perform these actions.

Recommendation 9: List All Security Assumptions

Since Marmot integrates different protocols, it is necessary to explicitly list the security assumptions carried over from those underlying protocols. Additionally, because Marmot modifies or extends certain protocol components, any newly introduced assumptions should be stated directly rather than implied. Finally, assumptions regarding the operations of subsystems (for example, the administration of authentication services and relays) are important to define as they can clarify the specific assurances Marmot is intended to provide.

Recommendation 10: Prevent Nonce Reuse for Encrypted Media

In MIP-04, the Marmot protocol defines functionality for sharing encrypted media files. The specification encrypts these files using `ChaCha20-Poly1305` with a key and nonce derived from the current epoch's group exporter secret. Since the secret remains constant for the duration of a group's epoch, nonce reuse may occur if the same file is sent twice by the same user within a single epoch. As a result, an attacker may establish an equality relationship between two separate encryptions of the same file, breaking ciphertext indistinguishability. We recommend modifying the specification to derive the nonce from a seed independently generated by a CSPRNG, in addition to the exporter secret.

Consulting Team

George Gkitsas, Security / Cryptography Researcher and Engineer

George holds an MEng in Electrical and Computer Engineering and has over a decade of experience in building and breaking security solutions across several verticals. His primary focus is on Cryptography and Vulnerability Research.

Anna Kaplan, Cryptography Researcher and Engineer

Anna is a mathematician and cryptographer, with experience working at Zcash Foundation and IBM Research. Besides being interested in new cryptographic advancements, she is also very passionate about communicating all things cryptography and privacy.

Miguel Quaresma, Security Researcher and Engineer

Miguel is a cryptography and security researcher with experience in cryptographic engineering and applied formal methods, currently pursuing his PhD at the Max Planck Institute for Security and Privacy. Previously, he worked as a cybersecurity analyst in the financial industry, where he was responsible for penetration tests and security analysis of infrastructures and applications.

About Least Authority

We believe that people have a fundamental right to privacy and that the use of secure solutions enables people to more freely use the Internet and other connected technologies. We provide security consulting services to help others make their solutions more resistant to unauthorized access to data and unintended manipulation of the system. We support teams from the design phase through the production launch and after.

The Least Authority team has skills for reviewing code in multiple Languages, such as C, C++, Python, Haskell, Rust, Node.js, Solidity, Go, JavaScript, ZoKrates, and circom, for common security vulnerabilities and specific attack vectors. The team has reviewed implementations of cryptographic protocols and distributed system architecture in cryptocurrency, blockchains, payments, smart contracts, zero-knowledge protocols, and consensus protocols. Additionally, the team can utilize various tools to scan code and networks and build custom tools as necessary.

Least Authority was formed in 2011 to create and further empower freedom-compatible technologies. We moved the company to Berlin in 2016 and continue to expand our efforts. We are an international team that believes we can have a significant impact on the world by being transparent and open about the work we do.

For more information about our security consulting, please visit https://leastauthority.com/security-consulting/.