Security
Research
Labs

OpenZeppelin runtime configuration template review

Hacking assessment report

V1.1, Aug 1st, 2024

Cayo Fletcher-Smith cayo@srlabs.de

Haroon Basheer haroon@srlabs.de

Regina Biro regina@srlabs.de

Abstract. This work describes the result of a thorough and independent security assurance audit performed by Security Research Labs on the OpenZeppelin runtime templates. Security Research Labs is a consulting firm that has been providing specialized audit services for Substrate-based blockchains in Polkadot ecosystem since 2019.

During this audit, OpenZeppelin provided access to relevant documentation and a revised threat model. The code of the OpenZeppelin EVM and generic runtime templates was verified to assure that the configuration and logic of the product is resilient to hacking and abuse.

The research team identified 2 high, 4 low and 1 info level severity issues in the configuration related to Cross chain messaging (XCM), runtime benchmarking for both generic and EVM runtime templates and incorrect values for runtime parameters. OpenZepplin team fixed 5 issues, 1 remain open and for 1 issue the risk was accepted.

In addition to mitigating the reported issues, Security Research Labs recommends further adherence to security standards in the Polkadot ecosystem through improved documentation of critical parameters, ensuring parameter value sanitization and validation. This will accelerate security awareness for new developers and improve the security posture for the runtime templates through accidental misconfigurations.

Content

1 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 2. 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 4 may not ensure all future code to be bug free.

2 Motivation and scope

OpenZeppelin [1] aims to provide a simplified entry into the Polkadot parachain ecosystem for substrate developers. The runtime configuration template developed by OpenZeppelin strives to provide a common framework and out-of-the-box solution for new developers to bootstrap and expand the Polkadot SDK for parachain development for EVM and generic runtime framework.

SRLabs conducted a generic runtime template audit for OpenZeppelin in April 2024 and published its final report [2]. In this follow-up engagement, the code assurance team focused on finding security vulnerabilities primarily stemming from misconfigurations in the EVM runtime construction library and performed a second round of audit for the generic template.

Component	Scope	Reference
EVM runtime template	Initial audit	[3]
Generic runtime template	Retest	[4]

Table 1 Runtime template audit scope

Security Research Labs (SRLabs) collaborated with the Parity AppSec team to conduct an in-depth review of the threat model document developed by OpenZeppelin [5]. An independent assessment into the runtime configuration and constructions was also conducted by the SRLabs auditors with their expertise in security auditing for various parachains and supporting secure development of Polkadot-SDK libraries. The OpenZeppelin runtimes were assessed for potential security flaws, realistic attack scenarios and adherence to security best practice in the Polkadot ecosystem.

During the assessment of the runtime templates, security issues were communicated to the OpenZeppelin and Parity AppSec team through a dedicated Element channel.

3 Baseline Assurance

3.1 Findings summary

During the analysis of the OpenZeppelin runtime templates, Security Research Labs identified 2 high, 4 low and 1 info-severity issues (7 in total), which are summarized in Table 1.

Issue	Template	Severity	Status
EVM runtime template waives XCM message delivery fee	EVM	High	Fixed
Incorrect runtime weights for treasury pallet	Generic	High	Fixed
ED of EVM template could suggest setting it to 0	EVM	Low	Risk accepted

Incorrect `BlockGasLimit` could lead to runtime panic	EVM	Low	Fixed
Use of depreciated `CurrencyAdapter` in the `transaction_payment`	EVM/ Generic	Low	Open
Misconfiguration of `MaxRemoteLockConsumers` in `pallet_xcm`	EVM/ Generic	Low	Fixed
Missing documentation undermines the template's use-case	EVM/ Generic	Info	Fixed

Table 2 Code audit issue summary

3.2 Detailed findings

3.2.1 EVM runtime template waives XCM message delivery fees

Attack scenario	EVM runtime template waives XCM message delivery fee
Location	evm-template/runtime/src/configs/xcm_config.rs
Attack impact	An attacker can cause network congestion causing delay in message deliveries or discards and storage exhaustions
Severity	High
Status	Fixed [6]

To avoid congestion and spamming of messages in the network via XCM, a fee is charged for message delivery to ensure fairness and optimal message delivery time amongst all the participants. This fee is configured in the runtime construct through use of `FeeManager` type.

In the EVM template for XCM runtime configuration for OpenZeppelin, have set the `FeeManager` [7] to the Rust unit type “`()`”. In the Polkadot-SDK, this implementation effectively will waive the delivery fees [8].

Thereby all fee-based congestion control mechanisms become ineffective since the fees are not actually charged. An attacker can cause congestion, possibly leading to long delays in message delivery, storage exhaustion and/or dropping of messages.

We recommend implementing a fee for XCM message delivery and follow the fix implemented in the OpenZeppelin generic template [9].

3.2.2 Incorrect runtime weights for treasury pallet

Attack scenario	Incorrect runtime weights for treasury pallet
Location	generic-template/runtime/src/configs/mod.rs
Attack impact	Extrinsic weights in treasury pallet are not dependent on the actual runtime configuration, this could lead to underweight extrinsics
Severity	High
Status	Fixed [10]

OpenZeppelin’s runtime template constructs runtime logics with FRAME pallets in its configuration. Appropriate benchmarking is required for the pallets to ensure its actual runtime performance. This will effectively transform into charging adequate fee for extrinsics execution.

For example, the runtime weights for the pallet treasury [11] in the EVM runtime template are configured using the default `SubstrateWeight`.

As these pallet extrinsic weights are not dependent on the actual runtime configuration and using default Substrate weights could leads to underweight extrinsics. An attacker may spam and bloat network storage freely using these underweight extrinsics

The `WeightInfo` of the treasury pallet should be configured using actual runtime benchmarks. A best practice example for configuring the weights of the treasury pallet can be found in the Kusama runtime implementation [12].

3.2.3 Existential deposit of EVM template could suggest setting it to 0

Attack scenario	Existential deposit configuration is suggested to be zero
Location	evm-template/runtime/src/constants.rs
Attack impact	Existential deposit of 0 could lead to spamming through creating a high number of accounts and thereby slowing down the chain operation
Severity	Low
Status	Risk accepted [13]

The minimum balance that an account must maintain to be considered a valid active account in Polkadot is `EXISTENTIAL_DEPOSIT` (ED). If an account’s balance falls below this threshold, all its funds are lost, and the account is removed by the network.

In the EVM template, ED is set [14] to zero for no benchmark feature build. A new runtime developer may accidentally set the ED value to zero. This could lead to an attacker creating multiple free accounts, occupying network resources and slowing down the chain. Allowing an account to remain active with zero ED could also lead to a spam attack.

It is recommended to remove the ED value of zero from the runtime template and document in inline comment that its value must be above zero. Every parachain will have its own threshold for ED, either in their native token or DOT. Furthermore, we recommend explaining these two configurations in the inline comment for the ED parameter in the runtime template. This will benefit new runtime developer by helping them avoid misconfiguring this safety-critical parameter.

A guideline on choosing a suitable ED value may be provided as below:

$$ED = Min\ Number\ of\ redundant\ collators \times Storage\ duration\ (years) \times Storage\ cost\ per\ year$$

3.2.4 Incorrect` parameter could lead to runtime panic

Attack scenario	Misconfiguration of dependent critical parameters in runtime
Location	evm-template/runtime/src/configs/mod.rs

Attack impact	Runtime crash or wraparound of gas limit ratio causing runtime panics
Severity	Low
Status	Fixed [15]

The `GasLimitPovSizeRatio` [16] is the ratio of the amount of Gas (`BlockGasLimit`) to the Proof Size (`MAX_POV_SIZE`) for a block. The `GasLimitPovSizeRatio` is used in `GasWeightMapping` [17] to obtain the weight of the corresponding proof size.

In the EVM runtime template, `GasLimitPovSizeRatio` may reach a large value or wrap to zero causing runtime crash. This can be enabled through setting the dependent parameter `BlockGasLimit` to zero or Max value in the runtime.

Ensure that `GasLimitPovSizeRatio` is bounded within `U64` data following the implementation in Frontier template from Polkadot [18]. Also consider documenting the risk of deviating outside the range such that the runtime developer is aware of potential risk from misconfiguring this value.

3.2.5 Usage of depreciated `CurrencyAdapter` in `transaction_payment`

Attack scenario	Usage may result in unexpected behavior and future edge-cases.
Location	runtime/src/configs/mod.rs
Attack impact	There may be inconsistencies in the computation of transferable balances, and future undefined behavior.
Severity	Low
Status	Open [19]

The runtime configuration for `pallet_transaction_payment` relies on the deprecated (since `v1.6.0`) `CurrencyAdapter [20].`

The discrepancy this introduces may result in unexpected behavior and inconsistencies when calculating transferable balances. We recommend migrating to the supported `FungibleAdapter` and `Fungible` traits.

3.2.6 Misconfiguration of `MaxRemoteLockConsumers` in `pallet_xcm`

Attack scenario	Misconfiguration may result in unintended side-effects and failed assumptions
Location	runtime/src/configs/xcm_config.rs
Attack impact	Unintentional behavior may be present in `pallet_xcm`
Severity	Low
Status	Fixed [21]

In `pallet_xcm::Config` both `MaxLockers` [22] and `MaxRemoteLockConsumers` [23] types have been set to the constant `MaxLockers: u32 = 8.`

The standard configuration of `MaxRemoteLockConsumers` is zero [24], which is declared in an unused constant `MaxRemoteLockConsumers: u32 = 0`

Based on this analysis and the lack of documentation around the deviation from the standard configuration, we assume this `MaxRemoteLockConsumers` was misconfigured to `8` not `0`.

This may result in unintended side-effects that are unknown to users implementing the template. If this configuration was intentional the rationale behind it should be explicitly stated.

We recommend configuring `MaxRemoteLockConsumers` to the declared constant `MaxRemoteLockconsumers: u32 = 0.`

3.2.7 Missing documentation undermines the templates' use-case

Attack scenario	Novice substrate developers may unknowingly alter sound security design due to ambiguity of rationale
Location	runtime/src/configs/
Attack impact	Configuration alteration could result in preventable vulnerabilities being re-opened.
Severity	Info
Status	Fixed [25]

Throughout the runtime configurations design rationale is frequently undocumented. Considering the runtime templates' use-case, to offer an entry point for secure substrate development, we believe the lack of documentation negatively impacts these intentions by introducing design ambiguity.

We recommend documenting all critical configurations to outline: the current design rationale; the security considerations for modification; and disclaimers around in-compatible modifications.

Specifically, we have identified the following key areas that require specific justification, although more likely exist:

1. In `xcm_executor` the mutual exclusivity of `IsReserve` [26] and `IsTeleport` [27] should be documented, alongside the importance of non-null `FeeManager` [28].

2. In `cumulus_pallet_xcmp_queue` `PriceForSiblingParachainDelivery` [29] the ramifications of null configuration should be described.

3. The rationale and risk of not including a non-zero `ExistentialDeposit` [30] in `pallet_balances` should be fully described.

4. The configuration options for `FeeMultiplierUpdate` [31] in `pallet_transaction_payment` should be explained (e.g `SlowAdjusting` / `FastAdjusting`).

5. All pallets `WeightInfo` configuration should have a disclaimer to perform benchmarking in the context of the user's specific runtime.

6. The process for calculating `GasLimitPovSizeRatio` [16] in `pallet_evm` should be outlined.

4 Evolution suggestions

To ensure that the runtime template is secure against known and yet undiscovered threats alike, the auditors recommend considering the evolution suggestions and best practices described in this section.

4.1 Improve threat model created for the runtime template

The threat document v2.0 shared by OpenZeppelin had minor improvements from the previous audit by identifying more relevant attack scenarios and their corresponding impact concerning the Polkadot runtime template. However, the document can be improved further for effective threat mapping to topics such as Assets, Actors, and Entry Points.

For instance, a table that maps each identified threat to its corresponding actors, the assets that will be compromised if exploited, and the possible entry points from the runtime template would be beneficial. This mapping will help novice developers enhance their security knowledge of the runtime, visualize the complete attack path, and reason about their custom configuration parameters.

Attackers' motivations to exploit a risk should be linked to the damage on the blockchain assets. The assets identified in the threat model span disparate classes such as CIA triad, XCM, tokens, and smart contracts. Asset categories can be further broken down into concrete topics such as components/pallets and motivation. For components/pallets, blockchain-critical systems such as XCM, Consensus, and Governance, which are configured through the runtime template, should be listed. As for motivation, metrics such as “Financial gain” or the CIA triad can be used. A table mapping an identified threats to the components and motivation could improve security awareness and the implications of misconfigurations in the template.

The threats identified in Table 8 of the document, such as unsafe pallet and smart contract usage and unsafe external dependencies, are closely related to attacks realized through supply chain threats. More specific threats related to runtime template misconfigurations that may pose risks such as executing free XCM messages, double spending tokens, manipulating on-chain voting, and breaking consensus should be considered.

Once the runtime threats are identified, they should be mapped to the corresponding attacks that can enable them. For instance, the current mapping in Table 8 maps attacks to the mitigation strategies and serves as a history of identified development bugs linked to the corresponding mitigation strategy. Consider adopting a standard threat modeling framework such as STRIDE, PASTA, etc. to enhance the threat modeling process. This will help define common terminology for validating identified threats and assist in developing new threats as the codebase evolves.

4.2 Improve documentation of runtime parameters

Generic and EVM runtime templates are meant to help with bootstrapping new runtime developers to accelerated development. Hence, it's critical to reason about and document pallet configurations and their values, especially when configuring parameters that have security relevance. This will ensure developers gain a deeper understanding of the inner workings of the Polkadot-SDK and effectively navigate its complexities through the runtime template.

We recommend expanding the existing documentation within codebase and external docs [32] to include more detailed explanations of configuration parameters, accepted range of values and the potential security pitfalls of associated misconfigurations.

Specific instances of inadequate documentation in the runtime template are outlined in the following subsections.

4.2.1 Hardcoded pallet number configurations in runtime macro

The configuration of the `construct_runtime!` macro has hardcoded pallet number configurations [33] without documentation. A runtime developer may deploy additional custom pallets depending on their business logic. Create a placeholder to define custom pallets and specify the range of pallet numbers that can be used for custom pallets. This will ensure standardization of pallet numbers used in the templates and improve maintenance as custom pallets are added to the runtime by the developer. As these pallet numbers are optional, explicitly documenting them will avoid incorrect configurations or default to auto resolve which may increase erroneously if the pallet numbers are duplicated.

4.2.2 OpenGov configuration should be documented

OpenGov is a framework for providing stakeholders of the network to contribute to the chain evolution and participate in decision making process. It enables transparent decision making amongst token holders to propose, vote and implement proposals. The parameters for OpenGov can be configured via the runtime.

The governance curve parameters should be documented to include a description of their purpose; alternative configurations; and associated security concerns. We recommend incorporating some disclaimer expressing the centralization risks around steep curves, specifically stating that: well-funded actors may perform root calls in short periods of time resulting in a disproportionate distribution of governance.

In Figure 1 and 2, the current governance parameters such as `min_approval` and `min_support` are plotted against the block time in hours. It was observed that the curves are not steep; for instance, after 50 hours have elapsed any root call referenda can be passed with less than 50% support. Even though this parameter is adopted from Polkadot, the configuration of these parameters should be explained in the template to aid developers if they decide to customize the referenda.

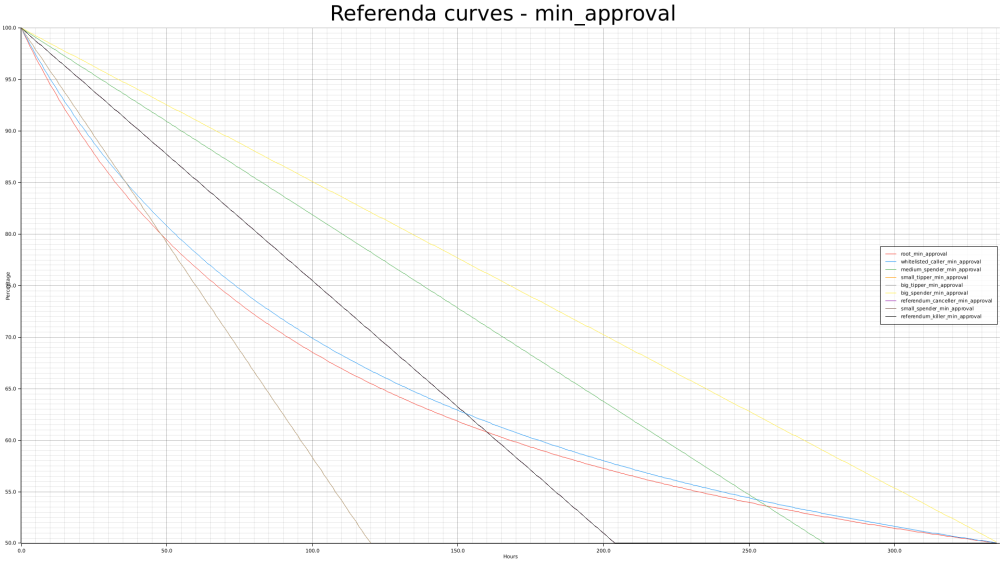

Figure 1 OZ Runtime template Governance minimum approval plot

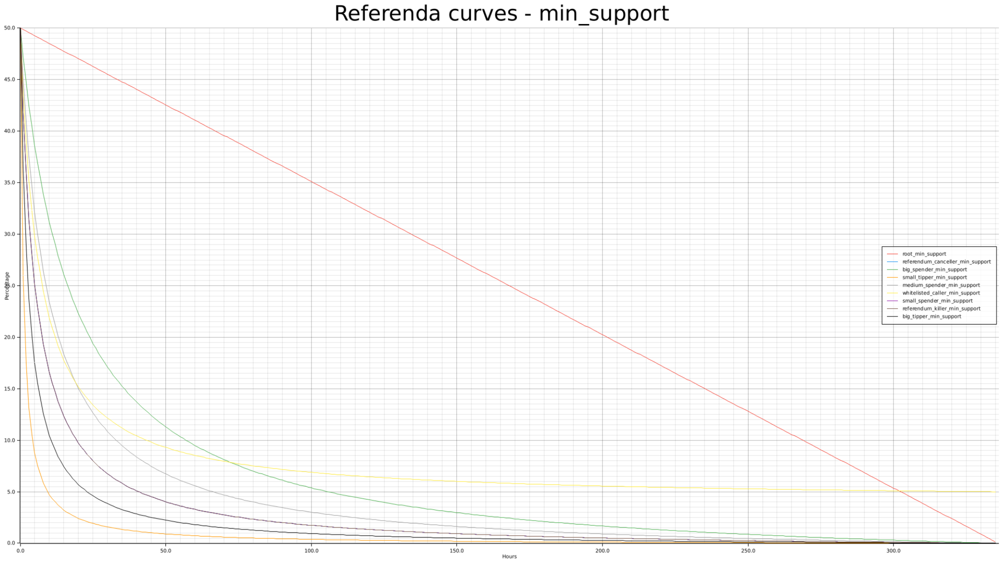

Figure 2 OZ Runtime template Governance minimum support plot

4.2.3 Provide detailed guidelines for configuring critical pallets

The documentation for configuring critical pallets that involve economics and block production, such as `pallet_balances`, `pallet_aura` should be expanded. As highlighted in 3.2.7, we recommend providing justification and pointers through inline documentation for parameter ranges and risk of deviating from standard values for these critical pallets. It is also encouraged to provide warnings through inline comments for configuration value that should not be changed by the runtime developer.

4.2.4 Forked repositories in the dependency should be highlighted

In the threat document, the current mitigation strategy for forking unsafe repositories is to adopt only repositories maintained by Parity. It is also indicated in the document that some unsafe dependencies cannot be avoided due to the lack of alternative safe implementations. The documentation for unsafe dependency usage is currently listed in the threat document, which may create decoupling between codebase and the documentation. It is recommended to list unsafe upstream dependency and provide justification through inline comments in `Cargo.toml` file or related config files/README explaining their intended usage, risks associated, and safety implemented in place if any within the codebase.

4.2.5 Document deviation from Polkadot standard runtime configurations

In conjunction with the recommendations expressed in section 3.2.7, we further emphasize the importance of integrating meaningful in-line documentation of runtime configuration parameters that:

- Have associated misconfiguration security risks
- Deviate from standard Parity configurations templates
- Are domain specific (e.g. Frontier) configurations.

5 Bibliography

[1] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-template/tree/c9a2c76a9db2e114eecaeba07195f9c2bdfaa094.

[2] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/tree/main/generic-template/audits.

[3] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/tree/main/evm-template.

[4] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/tree/main/generic-template.

[5] [Online]. Available: https://docs.google.com/document/d/1Hwz1sxp73RLBnsnbYMDGSWyS6Kj7p7lpjumgCkHifqg/edit?usp=sharing.

[6] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/pull/239.

[7] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-template/blob/c9a2c76a9db2e114eecaeba07195f9c2bdfaa094/runtime/src/xcm_config.rs#L153.

[8] [Online]. Available: https://github.com/paritytech/polkadot-sdk/blob/64660ee8d2e0d0ab4be3d416342463aaa6e168bd/polkadot/xcm/xcm-executor/src/traits/fee_manager.rs#L55-L57.

[9] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/generic-template/runtime/src/configs/xcm_config.rs#L159.

[10] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/pull/240.

[11] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/generic-template/runtime/src/configs/mod.rs#L588.

[12] [Online]. Available: https://github.com/peaqnetwork/polkadot/blob/016dc7297101710db0483ab6ef199e244dff711d/runtime/kusama/src/lib.rs#L810.

[13] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/issues/195.

[14] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/constants.rs#L17.

[15] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/pull/246.

[16] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/mod.rs#L596.

[17] [Online]. Available: https://github.com/polkadot-evm/frontier/blob/d200959a96e4b02eace07a7fc54eea28b5356f8b/frame/evm/src/lib.rs#L782-L786.

[18] [Online]. Available: https://github.com/paritytech/frontier-parachain-template/blob/ac10a6a534ea3a6e8f3a433f6ec1e08a7c8b76db/runtime/src/lib.rs#L739.

[19] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/issues/251.

[20] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/mod.rs#L329.

[21] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/pull/268.

[22] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/xcm_config.rs#L228.

[23] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/xcm_config.rs#L229.

[24] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/xcm_config.rs#L218.

[25] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/pull/250.

[26] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/xcm_config.rs#L159.

[27] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/xcm_config.rs#L160.

[28] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/xcm_config.rs#L155.

[29] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/mod.rs#L410.

[30] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/mod.rs#L307.

[31] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/configs/mod.rs#L327.

[32] [Online]. Available: https://docs.openzeppelin.com/substrate-runtimes/1.0.0/runtimes/generic.

[33] [Online]. Available: https://github.com/OpenZeppelin/polkadot-runtime-templates/blob/be0abb6bf1a42911867843e115d2a029ab26cbdc/evm-template/runtime/src/lib.rs#L175.