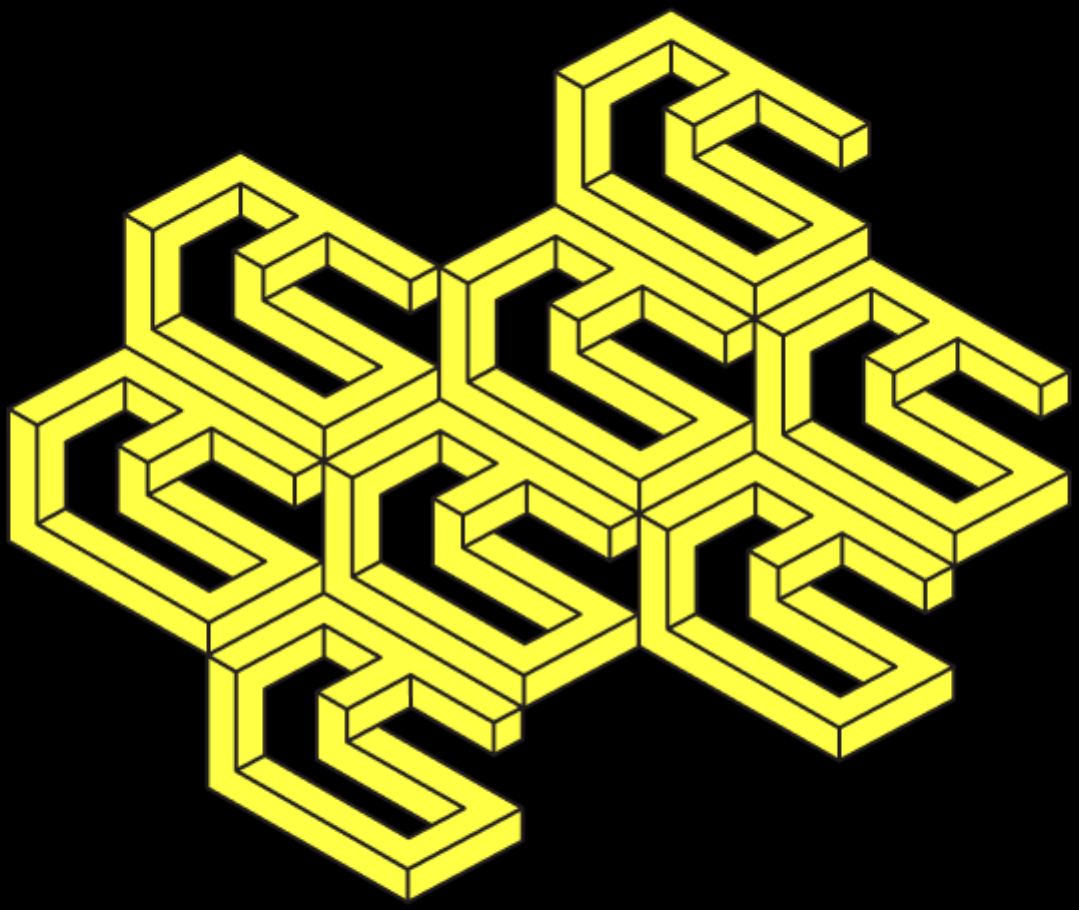

Propy Payment Smart Contract Review | February 2024

by ChainSafe Systems | February 2024

Table of contents

1. Introduction

Date	Auditor(s)
February 2024	Oleksii Matiiasevych, Tanya Bushenyova

Propy requested **ChainSafe Systems** to perform a review of the payment smart contracts. The contracts can be identified by the following git commit hash: `46752febdd0b893ac1ce28fb58918524eb154d64`

There is 1 contract in scope (`PaymentPRO.sol`).

Defining Severity

Each finding is assigned a severity level.

Note	Notes are informational in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
Optimization	Optimizations are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
Minor	Minor issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
Major	Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
Critical	Critical issues are directly exploitable security vulnerabilities that need to be fixed.

Referencing updated code

Resolved	The finding has been acknowledged and the team has since updated the code.
Rejected	The team dismissed this finding and no changes will be made.

Disclaimer

The review makes no statements or warranties about the utility of the code, safety of the code, suitability of the business model, regulatory regime for the business model, or any other statements about the fitness of the contracts for any specific purpose, or their bug free status.

2. Executive Summary

There are **no** known compiler bugs for the specified compiler version (^0.8.0), that might affect the contracts' logic.

There were **0 critical**, **0 major**, **1 minor**, 8 informational/optimization issues identified in the initial version of the contracts. These issues were acknowledged by the team. Propy team intended to fix the issues later without any additional verification. We enjoyed working with the Propy team, and liked how open and engaged they were in the discussion throughout the review.

3. Critical Bugs and Vulnerabilities

No critical issues were identified.

4. Line-by-line review

contracts/PaymentPRO.sol

L34 Note
In the `StrictPayment` structure storing both a reference (`paymentReference`) and a hashed reference (`paymentReferenceHash`) is redundant. Storing a hashed reference is not necessary because the hashed reference is the index for the mapping in which the `StrictPayment` struct is stored.

L41 Optimization
In the `StrictPayment` structure the field `exists` is redundant (and never read), other non-empty fields (for example, `tokenAddress`) can be used for checking existence if necessary.

L46 Note
`approvedPaymentTokens` and `approvedSweepingTokens` should contain the same set of tokens. Otherwise there could be a situation when some funds could not be swept. It could be useful to remove `approvedSweepingTokens` completely and just use `approvedPaymentTokens` everywhere.

L187 Note
`_tokenContract.transfer()` will not work for any token. It is recommended to utilize `safeTransfer` instead to assure support of all tokens. As long as it is used only for fully ERC20 compliant tokens, like Uniswap V2 LP, it will work correctly.

L224 Note
In the `makeDefaultPayment()` function the check that `defaultPaymentConfig.tokenAddress` is approved is probably redundant, that's possible only if the configuration is inconsistent.

L224 Optimization
In the `makeDefaultPayment()` function the `defaultPaymentConfig.tokenAddress` entry is read multiple times from storage. It could be saved in a local variable.

L237 Optimization
In the `makeStrictPayment()` function the `strictPayment.tokenAddress` entry is read multiple times from storage. It could be saved in a local variable.

L237 Note
In the `makeStrictPayment()` function the check for `approvedPaymentTokens` is probably redundant, the token was already checked during the creation of the `StrictPayment` entry.

L241 Minor

In the `makeStrictPayment()` function `strictPayment.complete` is not checked before execution. The same payment can be done multiple times.