HIPS Payment Group

Merchant Token

Security Assessment

March 17th, 2021

Audited By:
Alex Papageorgiou @ CertiK
alex.papageorgiou@certik.org
Reviewed By:
Camden Smallwood @ CertiK
camden.smallwood@certik.org

Disclaimer

CertiK reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has completed a round of auditing with the intention to increase the quality of the company/product's IT infrastructure and or source code.

Overview

Project Summary

Project Name	HIPS Payment Group - Merchant Token
Description	A typical ERC20 implementation.
Platform	Ethereum; Solidity, Yul
Codebase	N/A
Commits	N/A

Audit Summary

Delivery Date	March 17th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	March 16th, 2021 - March 17th, 2021

Vulnerability Summary

Total Issues	3
Total Critical	0
Total Major	0
Total Medium	0
Total Minor	1
Total Informational	2

Executive Summary

We were tasked with auditing the codebase of the merchant token located live at 0x55f3e422ed2e52347a3952dde18689518a7d7c9d.

The ERC-20 token does not deviate from the standard and is a simplistic implementation of it. Over the course of the audit we were able to pinpoint certain optimizations that can be made as well as a single vulnerability that is inherent to the ERC-20 standard and can only be exploited via user mishandling, however, we have included it as the token appears to be meant for payment methods and would greatly benefit from an alleviation to this issue.

The alleviated version of the token was deployed at 0xe66b3aa360bb78468c00bebe163630269db3324f including fixed for all three exhibits outlined by this report.

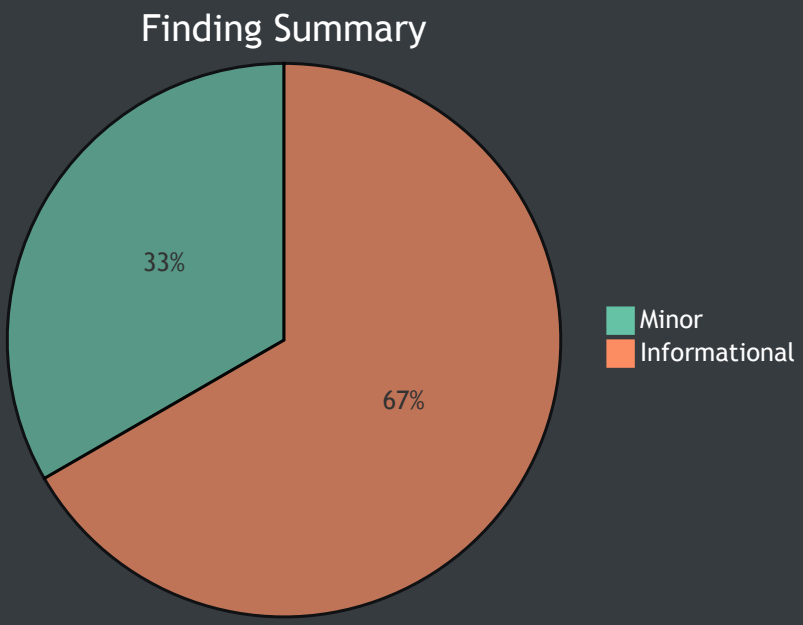
System Analysis

The token is straightforward and assigns the initial total supply to the creator of the contract at its construction. No other owner-controlled methods exist in the system.

Files In Scope

ID	Contract	Location
MTN	MerchantToken.sol	MerchantToken.sol

File Dependency Graph



Manual Review Findings

ID	Title	Type	Severity	Resolved
MTN-01	ERC-20 Inherent Approval Race Condition	Logical Issue	Minor	✓
MTN-02	Variable Mutability Specifiers	Gas Optimization	Informational	✓
MTN-03	Visibility Specifiers Missing	Language Specific	Informational	✓

MTN-01: ERC-20 Inherent Approval Race Condition

Type	Severity	Location
Logical Issue	Minor	MerchantToken.sol L65-L69

Description:

The ERC-20 standard contains a well-known flaw in its design whereby a race condition is introduced using its `approve` and `transferFrom` methods.

Recommendation:

While this would solely be exploitable in case of mishandling by the users, it should still be taken into consideration if the token is aimed to be utilized as a payment gateway. To this end, we advise that the `increaseApproval` and `decreaseApproval` functions are coded that prohibit this attack vector from being exploited.

Alleviation:

The HIPS team added an `increase` and `decrease` prefixed `Approval` function that is meant to allow users to circumvent the race-condition and state how much the allowance should increase or decrease by respectively, thus alleviating this issue.

MTN-02: Variable Mutability Specifiers

Type	Severity	Location
Gas Optimization	● Informational	MerchantToken.sol L34, L35, L36, L38, L44, L45, L46, L47

Description:

The linked variables are only assigned to once during the contract's `constructor` and are done so to value literals.

Recommendation:

We advise that the assignments are instead relocated to the variable declarations directly and that the variables are set to `constant` greatly optimizing the gas cost involved in utilizing them.

Alleviation:

All variables were set to `constant` according to our recommendation optimizing the gas cost of the contract.

MTN-03: Visibility Specifiers Missing

Type	Severity	Location
Language Specific	Informational	MerchantToken.sol L40, L41

Description:

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation:

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

Alleviation:

Visibility specifiers were added to the linked declarations thus avoiding compiler discrepancies and alleviating this issue.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.