Security Assessment

8Pay Core Contracts

May 21st, 2021

Table of Contents

Summary

This report has been prepared for 8Pay's core smart contracts in order to discover issues and vulnerabilities in the source code of the smart contracts, as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from minor to informational. We recommended addressing these findings to ensure a high level of security standards and industry practices. We suggested recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

All of the issues identified within the report were found to be resolved by the 8Pay team.

Overview

Project Summary

Project Name	8Pay Core Contracts
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/8pay/core-contracts
Commit	• <4a2037f3fd30f8b634247fac6e6ad39040cfb5c7> • <cdaee948f27a46821b9842b6c6437f037f17a04b>

Audit Summary

Delivery Date	Jun 16, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

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

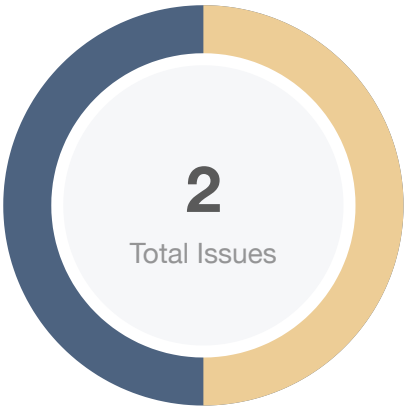
Audit Scope

ID	file	SHA256 Checksum
ACY	access/AccessControl.sol	8adf357537534210fbdebae3ccceb226962cb8ad19f38ecf23fa591a973a2cdb
FRC	billing-models/fixed-recurring/FixedRecurringConstants.sol	eea3655c978a3a43227902aa064a73649baefd029557bbb8521d9132bc443b10
FRP	billing-models/fixed-recurring/FixedRecurringPlans.sol	e3502ef61b23af9a82d2374f3ee5c8435101a1236ffd41dcaa9f8267aecbd7d5
FRD	billing-models/fixed-recurring/FixedRecurringPlansDatabase.sol	3f37a92694e1f13606fc871f57116e4a73b5b82003521a70dacb8d8f5aa4b713
FRS	billing-models/fixed-recurring/FixedRecurringSubscriptions.sol	7095a1684777ed34235a39417bd223abde5c4960f170afd5ed96002d4781a747
FSD	billing-models/fixed-recurring/FixedRecurringSubscriptionsDatabase.sol	8cbdaa5f9f2f601d0917d01baa7b1191a2b9335b38f562c7fe2ee3b9f07c8d6c
FRM	billing-models/fixed-recurring/FixedRecurringSubscriptionsManagement.sol	ef72b2871488067a369252539cdbf2a7e64e941de58fc4c7a8bf5526964f7488
ODC	billing-models/on-demand/OnDemandConstants.sol	87e1564d41ee3e88932ee9a8949f9e386e124f9067179a691b99f2336318ccc7
ODP	billing-models/on-demand/OnDemandPlans.sol	7dd322df691ef352307fef9aa2e24d72e0f1fe2f2375b859b9a0b7840485fd5b
ODD	billing-models/on-demand/OnDemandPlansDatabase.sol	1ac60544727d1ac722400754ba8bedeca9d83b80e0f86c6423c540bfb818903c
ODS	billing-models/on-demand/OnDemandSubscriptions.sol	53f44265a279f818fceee15bf6251a0552d6f9732386c5505ebec3d01e03831e
OSD	billing-models/on-demand/OnDemandSubscriptionsDatabase.sol	a569e54bb608711fbc41bfc0601430257a78d9ffc967360eeb17a0d5aeab4f1c
ODM	billing-models/on-demand/OnDemandSubscriptionsManagement.sol	fd28b8d23c2e7494015f6cfba2dc5608e1ca392da87e4aa81be68b276a397643
OTY	billing-models/one-time/OneTime.sol	808040fea01a6358c97402be78f0c9960ee2336a37748bf16ce3cc509df30b28
VRC	billing-models/variable-recurring/VariableRecurringConstants.sol	46ab4ef6aeb8a11e461edb9870b4a5f9a1bd0deec1bc76a855a902eae82bd2f0

ID	file	SHA256 Checksum
VRP	billing-models/variable-recurring/VariableRecurringPlans.sol	dc1b9fdc0aedd428d3f29672b440086d331a45b20f432ee593a2ad9d2a680566
VRD	billing-models/variable-recurring/VariableRecurringPlansDatabase.sol	d4a58d4714ff6de447cf40632acccf5c1924d3d7b5d70c916ad06c2920969d63
VRS	billing-models/variable-recurring/VariableRecurringSubscriptions.sol	789efe5b4feaaeea69fdff7353521ff0cfc42ec72c11a3be54a241cf4a4665e4
VSD	billing-models/variable-recurring/VariableRecurringSubscriptionsDatabase.sol	2c6e1a9d17af8fc1585415098465ec93f8fcd3f6246324f972fcc03f6722f727
VRM	billing-models/variable-recurring/VariableRecurringSubscriptionsManagement.sol	05d8bc5d04f9158d627cffe31af481d0c20b12e365d70e100f44870c8a9ab543
FPY	fees/FeeProvider.sol	3002a258a5a67c619e53ddb11715d1cc8ac672a3e2bd850f7e694bcb09e743c2
IFP	interfaces/IFeeProvider.sol	127d931cb4e5a972d5b9f765d26d5896c7ca3ba39d9dc87e0c8ee00931326fb9
IFR	interfaces/IFixedRecurringPlansDatabase.sol	3de9bf8d5fd2827d3fa2894c6f702023165535f0b5cdf4d97c111164cba21d81
IFS	interfaces/IFixedRecurringSubscriptionsDatabase.sol	fc3c02fd2baa202d921337861c68bc05a11d9a69ff097e3d4846414b1e711083
IOD	interfaces/IOnDemandPlansDatabase.sol	b4a0d3ab17d5c05ac94d6fd1c119ecf7288f00a8ff471a06039e2cbf0effd419
IOS	interfaces/IOnDemandSubscriptionsDatabase.sol	f1c1471986111b58752fc0bd093a2aceb0bd98b2b462d2794b95fc6a78f30539
ITR	interfaces/ITokensRegistry.sol	6bcab4d4b352d044d4d6736b8d880177ac29b0a0cef3533e7222b151119f1504
ITY	interfaces/ITransfers.sol	dcae9fe4ff5a948ae7a2c491e9f42e70c078f1330629afd18f6f263b528e1bce
IVR	interfaces/IVariableRecurringPlansDatabase.sol	79d1c53da1bc841a4936fe29fcd2fea431b6fd5558e74f90c8cfa95c6fb744ca
IVS	interfaces/IVariableRecurringSubscriptionsDatabase.sol	dc25c42ba529a1ef72228e7c0b6389b6a4953ac051626d97bbf1944ad9a48bc4
ARR	libraries/Arrays.sol	7b67d0e74fe5353d4a78bdba76882e49cbfb2b6bc1af08bd3bf3fc23e9831a26
ACM	mocks/AccessControlMock.sol	78a7bd3b9a26dff536e644f5c980a9f928b58fb85fe849160fbbebad9cb69b20
AMY	mocks/ArraysMock.sol	1ecd7d9d370e615bd817a8a43e65607078a566041258dab9d7d52e46e1aa67d1

ID	file	SHA256 Checksum
FPM	mocks/FeeProviderMock.sol	743fa669f867584b13b60ce308c3c402ac4ceec83b357ed92bdfdb103a52f990
MTY	mocks/MockToken.sol	cf53e99bdbd456022a9c72c2109d5be307b39ad836b6ad8b8161bf6bfbed48bb
DAT	storage/Database.sol	5713973212f55064fc1dd425e00bc694aa8500c49435bb5f9c26a271c78fabdd
TRY	tokens/TokensRegistry.sol	cc41ebbf3248d1b0b50a2b28b75b27d13a15ab1006a131d42026af37e80a0aa2
TRA	transfers/Transfers.sol	7cf788613e78e872ad17da1b3ce3ca3a8cfd3b889d6964997af98c34e89f1f96

Findings



Critical	0 (0.00%)
Major	0 (0.00%)
Medium	0 (0.00%)
Minor	1 (50.00%)
Informational	1 (50.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CON-01	Inefficient usage of `memory` references	Gas Optimization	Informational	Resolved
TRA-01	Raw address transfers in `_ethTransfer` function	Volatile Code	Minor	Resolved

CON-01 | Inefficient usage of `memory` references

Category	Severity	Location	Status
Gas Optimization	Informational		Resolved

Description

Any functions and variables which take/store `memory` references will copy `calldata` values into memory, which is not always efficient.

Recommendation

For gas optimization, we pointed out that the data location of all of the functions across all contracts accepting reference types can be changed from memory to calldata. For example, in the FixedRecurringPlans contract the functions createPlan, changeReceivers and _validateReceivers can have its reference type parameters' data location changed to calldata. Similarly, it will be cost effective to change data location from memory to calldata in the Database contract itself where the reference types are expected. One exception was the case where Array library is used with the memory type array.

Alleviation

The recommendation was found to be applied in commit 48d9caab7c7d5bf51346d3f5542a6a34d7aef645.

TRA-01 | Raw address transfers in `_ethTransfer` function

Category	Severity	Location	Status
Volatile Code	Minor	transfers/Transfers.sol: 311, 323, 327	Resolved

Description

The internal `_ethTransfer` function in the `Transfers` contract performed raw address transfers on lines 311, 323 and 327. After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer(amount)` for transferring ether due to only forwarding a hard-coded value of 2300 gas, which may not be enough if the recipient is a contract or if gas costs change.

Recommendation

Since the project already depends on the `@openzeppelin/contracts@4.1.0` node module, we recommended importing its `Address` library for the `address payable` type, followed by utilizing its `sendValue` function in place of the raw transfers on lines 311, 323 and 327. This will safely check the balance of the sending address, forward all available gas, as well as verify the result of the call for success:

```
import { Address } from "@openzeppelin/contracts/utils/Address.sol";
```

```
contract Transfers is ITransfers, Initializable, AccessControl {
    using Address for address payable;
    using SafeERC20 for IERC20;
```

```
payable(msg.sender).sendValue(msg.value);
```

```
payable(receivers[i]).sendValue(netAmount);
```

```
feeCollector.sendValue(totalFee);
```

Alleviation

The recommendation was found to be applied in commit d992851278d5048b60455b0518b97331067f2087.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings 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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.