SmartDec

PumaPayPullPayment Smart Contract Security Analysis

This report is public.

Published: October 12, 2018

Abstract

In this report, we consider the security of the PumaPay MasterPullPayment smart contract. Our task is to find and describe security issues in the smart contracts of the platform.

Disclaimer

The audit does not give any warranties on the security of the code. One audit cannot be considered enough. We always recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Besides, security audit is not an investment advice.

Summary

In this report, we have considered the security of PumaPay MasterPullPayment smart contract. We performed our audit according to the procedure described below.

The initial audit showed several low severity issues.

In the latest version of the code, all of these issues are fixed by the developer. However, new medium severity issue appeared due to the addition of new functionality. We discussed the issue with developers, and they provided us with the comments on how they plan to address it using off-chain backend. Comments are provided in the report below.

General recommendations

The contracts' code is of good code quality. The audits have shown no issues that endanger project security. However, we recommend taking into account that excessive Power of the executor should be carefully controlled with the off-chain backend.

Checklist

Security

The audit has shown no vulnerabilities.
Here by vulnerabilities we mean security issues that can be exploited by an external attacker. This does not include low severity issues, documentation mismatches, overpowered contract owner, and some other kinds of bugs.

Compliance with the documentation

The audit showed no discrepancies between the code and the provided documentation.

Tests

All the tests passed successfully

The text below is for technical use; it details the statements made in Summary and General recommendations.

Procedure

In our audit, we consider the following crucial features of the smart contract code:

1. Whether the code is secure.
2. Whether the code corresponds to the documentation (including whitepaper).
3. Whether the code meets best practices in efficient use of gas, code readability, etc.

We perform our audit according to the following procedure:

- automated analysis
  - we scan project's smart contracts with our own Solidity static code analyzer SmartCheck
  - we scan project's smart contracts with several publicly available automated Solidity analysis tools such as Remix and Solhint
  - we manually verify (reject or confirm) all the issues found by tools
- manual audit
  - we manually analyze smart contracts for security vulnerabilities
- report
  - we reflect all the gathered information in the report
  - we check the issues fixed by the developers

Checked vulnerabilities

We have scanned PumaPay MasterPullPayment smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered (the full list includes them but is not limited to them):

- Reentrancy
- Front Running
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Limit and Loops
- Locked money
- Integer overflow/underflow
- Unchecked external call
- ERC20 Standard violation
- Authentication with tx.origin
- Unsafe use of timestamp
- Using blockhash for randomness
- Balance equality
- Unsafe transfer of ether
- Fallback abuse
- Using inline assembly
- Short Address Attack
- Private modifier
- Compiler version not fixed
- Style guide violation
- Unsafe type deduction
- Implicit visibility level
- Use delete for arrays
- Byte array
- Incorrect use of assert/require
- Using deprecated constructions

Project overview

Project description

In our analysis we consider PumaPay smart contract code:

1. version 1, commit: badb0ec7b30821f3d34f8acab2876ac065dcc238
2. version 2, commit: 30d549e2eeea1de569d22ad570a0abadf251035a
3. version 3, commit: 7998098f29613ba16172518de5d0b676ddba3762

All the outputs in Appendix are performed for the latest version of the code.

Project architecture

For the audit, we were provided with the following file:

- **PumaPayPullPayment.sol** (**MasterPullPayment.sol** in previous versions of the code)
- The file successfully compiles with `truffle compile` command (see Compilation output in Appendix)
- The file successfully passes all the tests (see Tests output in Appendix)

The total LOC of audited Solidity sources is 249.

Automated analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Remix. All the issues found by tools were manually checked (rejected or confirmed).

True positives are constructions that were discovered by the tools as vulnerabilities and can actually be exploited by attackers or lead to incorrect contracts operation.

False positives are constructions that were discovered by the tools as vulnerabilities but do not consist a security threat.

Cases when these issues lead to actual bugs or vulnerabilities are described in the next section.

Tool	Rule	True positives	False positives
Remix	Gas requirement of function high: infinite		6
	Potential Violation of Checks-Effects-Interaction pattern	1	
Total Remix		1	6
SmartCheck	Private Modifier Dont Hide Data		4
	Revert Require	2	
	Safemath	1	
	Timestamp Dependence		2
	Underflow Overflow		1
Total SmartCheck		3	7

Solhint	Avoid to make time-based decisions in your business logic		4
Total Solhint			4
Total Overall		**4**	**17**

Manual analysis

The contract was completely manually analyzed, its logic was. Besides, the results of the automated analysis were manually verified. All confirmed issues are described below.

Critical issues

Critical issues seriously endanger smart contracts security. We highly recommend fixing them.

The audit showed no critical issues.

Medium severity issues

Medium issues can influence smart contracts operation in current implementation. We highly recommend addressing them.

Potential money loss

In the latest version of code, new functionality is added: smart contract transfers 1 ETH to executor when his balance is lower than 0.01 ETH. However there is a vulnerability: the executor can transfer granted ETH from this account after every transaction, so protocol will systematically grant ETH to the executor.

According to the documentation,

> "The executor(s) is an address owned by the association governing the smart contract."

However, if executor's private keys become compromised, this brings significant threat to the whole project. We recommend implementing additional logic for limiting transfers to executors or checking if ETH is used properly.

For example, such check can be implemented with using of off-chain backend tool. It could observe executors activity on blockchain via checking their transactions and raise warnings in case of any unfair spending of ETH.

- *The issue appeared in version 2 of the code.*
- *Comment from the developers:*

"Regarding the issue of Potential Money Loss we are aware of such situation and we have the following in place:

1. Tool that monitors the executor and owner addresses
a. In case they transfer ETH we get notified
b. We transfer the ownership to a different address
c. We remove the existing executors and add new ones
2. We have contingency plan in place to notify the end users

Even in the event of being compromised, existing users of the smart contract will not be affected, i.e. execution of pull payments will still be executed. Part of the contingency plan is to deploy a new smart contract with new owner/executors to allow for future pull payments registration/cancellation."

Low severity issues

Low severity issues can influence smart contracts operation in future versions of code. We recommend taking them into account.

`revert()` vs `require()`

`revert()` is used at several places:

- line 210:

```
if (!isValidRegistration(v, r, s, _client, _beneficiary,
pullPayments[_client][_beneficiary])) revert();
```

- line 246:

```
if (!isValidDeletion(v, r, s, _paymentID, _client,
_beneficiary)) revert();
```

We recommend using `require(condition);` instead of `if (!condition) {revert();}` to improve code readability and transparency.
The issue has been fixed by the developers and is not present in version 2 and version 3 of the code.

Violation of Checks-Effects-Interaction pattern

The Checks-Effects-Interaction pattern is violated at lines 286-288:

```
token.transferFrom(_client, msg.sender, amountInPMA);
pullPayments[_client][msg.sender].lastPaymentTimestamp = now;
```

In this case, the CEI violation does not lead to an actual vulnerability. However, we highly recommend following best practices including Checks-Effects-Interactions pattern since it helps to avoid many serious vulnerabilities.
The issue has been fixed by the developers and is not present in version 2 and version 3 of the code.

Redundant code

1. The check at line 196 is redundant:

```
_initialPaymentAmountInCents >= 0
```

   The `_initialPaymentAmountInCents` variable is of `uint256` type and, thus, always non-negative.
   The issue has been fixed by the developers and is not present in version 3 of the code.

2. Moreover, line 12 is redundant:

```
using SafeMath for uint256;
```

- *The issue appeared in version 2 of the code.*
- *The issue is not present in version 3 of the code.*

We highly recommend removing redundant code in order to improve code readability and transparency and decrease cost of deployment and execution.

OpenZeppelin version

The version of **OpenZeppelin** library used in the project is not fixed (**package.json**, line 19):

```
"openzeppelin-solidity": "^1.12.0"
```

Future versions of **OpenZeppelin** library can include some critical changes. Such changes can lead to behaviour that the developers have not foreseen.
We recommend specifying the exact version of the **OpenZeppelin** library in **package.json**.
The issue has been fixed by the developers and is not present in version 2 and version 3 of the code.

Use of SafeMath

SafeMath library is used at line 318 in version 1 of the code:

```
return
ONE_ETHER.mul(DECIMAL_FIXER).mul(_fiatAmountInCents).div(excha
ngeRates[_currency]).div(FIAT_TO_CENT_FIXER);
```

Using of **SafeMath** do not lead to any vulnerabilities in the current line. However, we recommend replacing **SafeMath** at this line with the ordinary arithmetic operations and the corresponding checks in order to reduce gas consumption.

The issue has been fixed by the developers and was not present in version 2 of the code.

- *Comment from the developers for version 3:*

"We have switched back to using the SafeMath library even though it is more expensive."

This analysis was performed by SmartDec.

Evgeniy Marchenko, Lead Developer
Alexander Seleznev, Chief Business Development Officer
Boris Nikashin, Project Manager
Alexander Drygin, Analyst
Kirill Gorshkov, Analyst
Igor Sobolev, Analyst

Sergey Pavlin, Chief Operating Officer 

October 12, 2018

Appendix

Compilation output

```
Compiling ./contracts/Migrations.sol...
Compiling ./contracts/PumaPayPullPayment.sol...
Compiling ./contracts/PumaPayToken.sol...
Compiling ./contracts/PumaPayVault.sol...
Compiling ./contracts/TimeVesting.sol...
Compiling ./contracts/TokenMultiSig.sol...
Compiling ./node_modules/openzeppelin-
solidity/contracts/math/SafeMath.sol...
Compiling ./node_modules/openzeppelin-
solidity/contracts/ownership/Ownable.sol...
Compiling ./node_modules/openzeppelin-
solidity/contracts/token/ERC20/BasicToken.sol...
Compiling ./node_modules/openzeppelin-
solidity/contracts/token/ERC20/ERC20.sol...
Compiling ./node_modules/openzeppelin-
solidity/contracts/token/ERC20/ERC20Basic.sol...
Compiling ./node_modules/openzeppelin-
solidity/contracts/token/ERC20/MintableToken.sol...
Compiling ./node_modules/openzeppelin-
solidity/contracts/token/ERC20/StandardToken.sol...
Writing artifacts to ./build/contracts
```

Tests output

```
Contract: PumaPay Pull Payment Contract
    Deploying
      ✓ PumaPay Pull Payment owner should be the address that
was specified on contract deployment
      ✓ PumaPay Pull Payment token should be the token address
specified on contract deployment
      ✓ PumaPay Pull Payment deployment should revert when the
token is a ZERO address
    Add executor
      ✓ should set the executor specified to true
```

```
      ✓ should transfer ETHER to the executor account for
paying gas fees
      ✓ should revert when the executor is a ZERO address
      ✓ should revert when the adding the same executor
      ✓ should revert if NOT executed by the owner
    Remove executor
      ✓ should set the executor specified to false
      ✓ should revert when the executor is a ZERO address
      ✓ should revert when the executor does not exists
      ✓ should revert if NOT executed by the owner
    Set Rate
      ✓ should set the rate for fiat currency
      ✓ should set the rate for multiple fiat currencies
      ✓ should revert when not executed by the owner
      ✓ should allow everyone to retrieve the rate
      ✓ should emit a "LogSetExchangeRate" event
    Register Pull Payment
      ✓ should add the pull payment for the beneficiary in the
active payments array
      ✓ should revert when NOT executed an executor
      ✓ should revert when the pull payment params does match
with the ones signed by the signatory
      ✓ should emit a "LogPaymentRegistered" event
    Delete Recurring Payment
      ✓ should set the cancel date of the pull payment for the
beneficiaryOne to NOW
      ✓ should revert when NOT executed by an executor
      ✓ should revert when the payment for the beneficiary
does not exists
      ✓ should revert when the deletion pull payment params
does match with the ones signed by the signatory
      ✓ should emit a "LogPaymentCancelled" event
    Execute Single Pull Payment
      ✓ should pull the amount specified on the payment
details to the beneficiaryOne
      ✓ should update the pull payment numberOfPayments
      ✓ should update the pull payment nextPaymentTimestamp
      ✓ should update the pull payment lastPaymentTimestamp
      ✓ should revert if executed before the start date
specified in the payment
```

```
      ✓ should revert when executed twice, i.e. number of
payments is zero
      ✓ should revert when pull payment does not exists for
beneficiary calling the smart contract
      ✓ should emit a "LogPullPaymentExecuted" event
    Execute Recurring Pull Payment
      ✓ should pull the amount specified on the payment
details to the beneficiary
      ✓ should update the pull payment numberOfPayments
      ✓ should update the pull payment nextPaymentTimestamp
      ✓ should update the pull payment lastPaymentTimestamp
      ✓ should execute the next payment when next payment date
is reached
      ✓ should revert when if the next payment date is NOT
reached
      ✓ should allow the merchant to pull payments in case
they have missed few payments
      ✓ should allow the merchant to pull payments in case
they have missed few payments and the customer cancelled the
subscription
    Execute Recurring Pull Payment with initial amount
      ✓ should pull the initial amount specified on the
payment details to the beneficiary
      ✓ should pull the amount of the first payment specified
for the reccuring payment to the beneficiary after receiving
the initial payment
      ✓ should pull the amount of the second payment specified
for the reccuring payment to the beneficiary
      ✓ should set the intial payment amount to ZERO after
pulling it

  Contract: PumaPay Pull Payment Contract For Funding
    Set Rate
      ✓ should transfer ETH to the owner when its balance is
lower than 0.01 ETH and set the rate
    Add Executor
      ✓ should transfer ETH to the owner when its balance is
lower than 0.01 ETH
    Remove Executor
      ✓ should transfer ETH to the owner when its balance is
lower than 0.01 ETH
    Register Pull Payment
```

```
      ✓ should transfer ETH to the executor when its balance
is lower than 0.01 ETH
    Delete Pull Payment
      ✓ should transfer ETH to the executor when its balance
is lower than 0.01 ETH
```

Solhint output

```
PumaPayPullPayment.sol
   32:2   error    Line length must be no more than 120 but
current length is 137                          max-line-length
   33:2   error    Line length must be no more than 120 but
current length is 125                          max-line-length
   90:14  warning  Avoid to make time-based decisions in your
business logic                                 not-rely-on-time
   91:13  warning  Avoid to make time-based decisions in your
business logic                                 not-rely-on-time
   96:2   error    Line length must be no more than 120 but
current length is 122                          max-line-length
  124:5   error    Definitions inside contract / library must
be separated by one line                       separate-by-one-
line-in-contract
  126:5   error    Open bracket must be on same line. It must
be indented by other constructions by space  bracket-align
  132:33  warning  Code contains empty
block
      no-empty-blocks
  143:5   error    Definitions inside contract / library must
be separated by one line                       separate-by-one-
line-in-contract
  150:9   warning  Possible reentrancy vulnerabilities. Avoid
state changes after transfer                   reentrancy
  198:2   error    Line length must be no more than 120 but
current length is 179                          max-line-length
  214:5   error    Definitions inside contract / library must
be separated by one line                       separate-by-one-
line-in-contract
  267:2   error    Line length must be no more than 120 but
current length is 156                          max-line-length
  268:2   error    Line length must be no more than 120 but
current length is 138                          max-line-length
  294:63  warning  Avoid to make time-based decisions in your
business logic                                 not-rely-on-time
```

```
  322:5   error    Definitions inside contract / library must
be separated by one line                    separate-by-one-
line-in-contract
  329:2   error    Line length must be no more than 120 but
current length is 154                        max-line-length
  332:2   error    Line length must be no more than 120 but
current length is 144                        max-line-length
  334:2   error    Line length must be no more than 120 but
current length is 170                        max-line-length
  337:66  warning  Avoid to make time-based decisions in your
business logic                              not-rely-on-time
  351:2   error    Line length must be no more than 120 but
current length is 121                        max-line-length
  364:5   error    Definitions inside contract / library must
be separated by one line                    separate-by-one-
line-in-contract
  368:2   error    Line length must be no more than 120 but
current length is 122                        max-line-length

✖ 23 problems (17 errors, 6 warnings)
```

Solium output

```
PumaPayPullPayment.sol
 63:8 warning Provide an error message for require(). error-
reason
 68:8 warning Provide an error message for require(). error-
reason
 73:8 warning Provide an error message for require(). error-
reason
 78:8 warning Provide an error message for require(). error-
reason
 83:8 warning Provide an error message for require(). error-
reason
 88:8 warning Provide an error message for require(). error-
reason
 89:12 error Only use indent of 12 spaces. indentation
 89:12 warning Operator "&&" should be on the line where left
side of the Binary expression ends. operator-whitespace
 89:52 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 89:52 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
```

```
 90:13 warning Avoid using 'now' (alias to 'block.timestamp').
security/no-block-members
 91:12 warning Avoid using 'now' (alias to 'block.timestamp').
security/no-block-members
 94:12 warning In Binary Expressions that span over multiple
lines, expression on the right side of the operator (&&) must
be exactly 1 line below the line on which the left expression
ends. operator-whitespace
 105:8 warning Provide an error message for require(). error-
reason
 106:40 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 114:8 warning Provide an error message for require(). error-
reason
 127:8 warning Provide an error message for require(). error-
reason
 232:8 warning Provide an error message for require(). error-
reason
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 233:53 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 252:8 warning Provide an error message for require(). error-
reason
 292:8 warning Provide an error message for require(). error-
reason
 294:62 warning Avoid using 'now' (alias to
'block.timestamp'). security/no-block-members
 337:65 warning Avoid using 'now' (alias to
'block.timestamp'). security/no-block-members
 428:33 warning Operator "&&" should be on the line where left
side of the Binary expression ends. operator-whitespace
 434:15 error Only use indent of 12 spaces. indentation
 434:18 error Only use indent of 12 spaces. indentation
 434:21 error Only use indent of 12 spaces. indentation
```

```
 448:68 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 449:68 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 449:68 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace
 449:68 warning Single space should be either on both sides of
'&&' or not at all. operator-whitespace

✖ 4 errors, 34 warnings found.
```