SmartDec

PumaPay Smart Contracts Security Analysis

This report is public.

Published: August 1, 2019.

Abstract

In this report, we consider the security of the PumaPay project. 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 considered the security of PumaPay smart contracts. We performed our audit according to the procedure described below.

The initial audit showed no critical issues. However, one medium severity and a number of low severity issues were found. They do not endanger project security. Nevertheless, we highly recommend addressing them.

Some of the issues were fixed in the latest version of the code.

General recommendations

The contracts code is of medium code quality. The audit did not reveal any issues that endanger project security.

In addition, if the developers decide to improve the code, we recommend fixing Coverage issue. However, mentioned above is minor issue. It does not affect code operation.

Checklist

Security

The audit showed 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 pass without any issues, however, the coverage script fails due to an exception.

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
  - we categorize all the found security issues in accordance with the classification in order to identify developers' shortcomings
  - we check smart contracts logic and compare it with the one described in the documentation
  - we run tests
- report
  - we reflect all the gathered information in the report

Checked vulnerabilities

We have scanned PumaPay smart contracts 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 specification (`docs` folder from repository) and smart contracts' code (version on commit 5eb99b1a94d9e5d98873fb4338b97943b9821569).

The latest version of the code

After the initial audit, some fixes were applied and the code was updated to the latest version (commit c248be94bb00ac19eb0d53629a4698f86da3d3f9).

Project architecture

For the audit, we were provided with the truffle project. The project is an npm package and includes tests.

- The project successfully compiles with `truffle compile` command (see Compilation output in Appendix)
- The project successfully passes all the tests

The total LOC of audited Solidity sources is 817.

Automated analysis

We used several publicly available automated Solidity analysis tools. Here are the combined results of SmartCheck, Solhint, and Remix scanning. 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
SmartCheck	Overpowered role	1	
	Private modifier	20	
	Use of SafeMath	2	
	Hardcoded address		1
Total SmartCheck		23	1
Remix	Potential Violation of Checks-Effects-Interaction pattern		3
	Constant but potentially should not be		4
	Use of ”now”		4
Total Remix		0	11
Solhint	Avoid to make time-based decisions in your business logic		11
	Possible reentrancy vulnerabilities. Avoid state changes after transfer		2
Total Solhint		0	13
Total Overall		**23**	**25**

Manual analysis

The contracts were completely manually analyzed, their logic was checkedand compared with the one described in the documentation. Besides, the results of the automated analysis were manually verified. All the 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.

Overpowered owner

PumaPayPullPayment.sol, line 212:

```
function setRate(string memory _currency, uint256 _rate)
```

This function allows the owner of the contract to change the exchange rate at any time. Thus, the owner can change the rate when a user's payment is in process, which can lead to unexpected outcome of the payment for the user. **PumaPayPullPaymentV2.sol** contract also contains this issue, line 428:

```
function executePullPayment(address _customerAddress,
bytes32 _paymentID, uint256 _conversionRate)
```

Executor of `pullPayment` can change the rate while the payment is in process.

Moreover, since executors register and execute pull payments, they can postpone or ignore certain payments processing.

In the current implementation, the system depends heavily on the owner of the contract. In this case, there are scenarios that may lead to undesirable consequences for investors, e.g. if the owner's private keys become compromised. Thus, we recommend designing contracts in a trustless manner.

Comment from the developers: ”PumaPay always places our merchants-first approach in regards to our Pull Payment Protocol. When listing their products/services for purchase, merchants set their prices in fiat currency and not in PMA. It is a necessity to have the conversion rate placed on the smart contract, to ensure both merchant and customer are in agreement on the PMA price following conversion. As the initial PMA price is not listed by the merchant, the conversion rate placed on the smart contract, gives the final price in PMA that the customer will be paying.
In our V1 smart contracts, the conversion rate is a global variable that we refresh every 10 minutes, however in the V2 smart contracts, the conversion rate will be produced live at the moment of execution giving the most accurate rate available.
Currently, the executors of the transaction lifecycle are addresses owned by PMA and thus we are committed to ensure that we deliver on all registrations, cancellations and executions of all our Pull Payments.
It is important to note that we are developing a set of smart contracts with the aim of removing the ”overpowered owner” nature of our current smart contracts.”

Low severity issues

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

Incorrect check (fixed)

There are incorrect checks:

- **PumaPayPullPayment.sol**, lines 130, 263, 264:

```
require(_paymentID.length != 0, "Invalid deletion request -
Payment ID is empty.");
...
require(_ids[0].length > 0, "Payment ID is empty.");
require(_ids[1].length > 0, "Business ID is empty.");
```

- **PumaPayPullPaymentV2.sol**, lines 147, 163, 264-266:

```
require(_paymentID.length != 0, "Invalid deletion request -
Payment ID is empty.");
...
require(_paymentType.length > 0, "Payment Type is empty.");
...
```

```
require(_paymentDetails[0].length > 0, "Payment ID is
empty.");
require(_paymentDetails[1].length > 0, "Business ID is
empty.");
require(_paymentDetails[2].length > 0, "Unique Reference ID
is empty.");
```

The length of `bytes32` type is always 32. Therefore, we recommend implementing the following check instead:

```
bytes32 constant internal EMPTY_BYTES32 = "";
...
require(someVariable != EMPTY_BYTES32);
```

The issues have been fixed and are not present in the latest version of the code.

Coverage

The coverage script provided with the code failed (see Code coverage in Appendix). Testing is crucial for the code security. We highly recommend fixing the code so that the coverage script will run successfully.

Misleading comments (fixed)

The following comments are misleading:

1. According to the comment (**PumaPayPullPayment.sol**, line 535):

```
/// The minimum amount the owner/executors should always
have is 0.001 ETH
```

However, it is 0.15 ETH in the contract code, **PumaPayPullPayment.sol**, line 51:

```
uint256 constant private MINIMUM_AMOUNT_OF_ETH_FOR_OPERATORS
= 0.15 ether; /// min amount of ETH for owner/executor
```

2. According to the comment (**PumaPayPullPaymentV2.sol**, line 607):

```
/// The minimum amount the owner/executors should always
have is 0.001 ETH
```

However, it is 0.15 ETH, **PumaPayPullPaymentV2.sol**, line 56:

```
uint256 constant private MINIMUM_AMOUNT_OF_ETH_FOR_OPERATORS
= 0.15 ether; /// min amount of ETH for owner/executor
```

We recommend fixing these comments in order to avoid confusion and improve code readability.

The issues have been fixed and are not present in the latest version of the code.

Excessive gas consumption (fixed)

PumaPayPullPayment.sol, lines 380-384:

```
if (pullPayments[_customer][msg.sender].initialPaymentA-
mountInCents > 0)
{
    amountInPMA = calculatePMAFromFiat(
    pullPayments[_customer][msg.sender].initialPaymentA-
mountInCents,
    pullPayments[_customer][msg.sender].currency
    );
```

`pullPayments[_customer][msg.sender].initialPaymentAmountInCents` variable is read twice. We recommend reading this variable once and using addition `memory` variable in both places in order to reduce gas consumption.

The issue has been fixed and is not present in the latest version of the code.

Missing event (fixed)

There are no events emitted when ether is sent to the owner or to an executor. We recommend adding such an event and emitting it in all appropriate places.

The issue has been fixed and is not present in the latest version of the code.

Redundant check (fixed)

There is a redundant check, **PumaPayPullPaymentV2.sol**, line: 163-171:

```
require(_paymentType.length > 0, "Payment Type is empty.");
require(
  (
    _paymentType == TYPE_SINGLE_PULL_PAYMENT ||
    _paymentType == TYPE_RECURRING_PULL_PAYMENT ||
    _paymentType == TYPE_RECURRING_PULL_PAYMENT_WITH_INITIAL
||
    _paymentType == TYPE_PULL_PAYMENT_WITH_FREE_TRIAL ||
    _paymentType == TYPE_PULL_PAYMENT_WITH_PAID_TRIAL),
  "Payment Type provided not supported");
```

The first check is then duplicated in the second check. Thus, the first `require` statement is redundant and we recommend removing it.

The issue has been fixed and is not present in the latest version of the code.

Unused import (fixed)

There are unused imports:

- **PumaPullPayment.sol**, line 6

```
import "openzeppelin-
solidity/contracts/token/ERC20/ERC20Mintable.sol";
```

  ERC20Mintable contract is imported, however it is not used anywhere in the code.

- **PumaPayPullPaymentV2.sol**, line 4

```
import "openzeppelin-
solidity/contracts/token/ERC20/ERC20Mintable.sol";
```

  ERC20Mintable contract is imported, however it is not used anywhere in the code.

We recommend removing unused imports in order to improve code readability.

The issues have been fixed and are not present in the latest version of the code.

Notes

Private modifier

There are variables with `private` visibility modifier:

- **PumaPayPullPaymentV2.sol**, lines 49-63:

```
uint256 constant private DECIMAL_FIXER = 10 ** 10;
uint256 constant private FIAT_TO_CENT_FIXER = 100;
uint256 constant private OVERFLOW_LIMITER_NUMBER = 10 ** 20;
uint256 constant private ONE_ETHER = 1 ether;
uint256 constant private FUNDING_AMOUNT = 1 ether;
uint256 constant private MINIMUM_AMOUNT_OF_ETH_FOR_OPERATORS
= 0.15 ether;
bytes32 constant private TYPE_SINGLE_PULL_PAYMENT = "2";
bytes32 constant private TYPE_RECURRING_PULL_PAYMENT = "3";
bytes32 constant private
TYPE_RECURRING_PULL_PAYMENT_WITH_INITIAL = "4";
bytes32 constant private TYPE_PULL_PAYMENT_WITH_FREE_TRIAL =
"5";
bytes32 constant private TYPE_PULL_PAYMENT_WITH_PAID_TRIAL =
"6";
bytes32 constant private TYPE_SINGLE_DYNAMIC_PULL_PAYMENT =
"7";
```

- **PumaPayPullPayment.sol**, lines 45-51, 59:

```
uint256 constant private DECIMAL_FIXER = 10 ** 10;
uint256 constant private FIAT_TO_CENT_FIXER = 100;
uint256 constant private OVERFLOW_LIMITER_NUMBER = 10 ** 20;
uint256 constant private ONE_ETHER = 1 ether;
uint256 constant private FUNDING_AMOUNT = 1 ether;
uint256 constant private MINIMUM_AMOUNT_OF_ETH_FOR_OPERATORS
= 0.15 ether;
...
mapping(string => uint256) private conversionRates;
```

- **PayableOwnable.sol**, line 11:

```
address payable private _owner;
```

We recommend taking into account that, contrary to a popular misconception, anyone can see values of `private` variables in the blockchain.

This analysis was performed by SmartDec.

Boris Nikashin, Project Manager
Pavel Kondratenkov, Analyst
Alexander Drygin, Analyst

August 1, 2019

Appendix

Code coverage

```
Error: Error: Error: while migrating Migrations: Returned er
ror: VM Exception while processing transaction: invalid opco
de
    at Object.run (/home/pavelcore/projects/smartdec/pumapay
/june/smart-contracts/node_modules/truffle/build/webpack:/pa
ckages/truffle-migrate/index.js:84:1)
    at <anonymous>
    at process._tickCallback (internal/process/next_tick.js:
188:7)
Truffle v5.0.19 (core: 5.0.19)
Node v8.10.0

Istanbul coverage reports generated
Cleaning up...
Shutting down testrpc-sc (pid 18115)
Some truffle tests failed while running coverage
```

Compilation output

```
Compiling your contracts...
===========================
> Compiling ./contracts/Migrations.sol
> Compiling ./contracts/PumaPayPullPayment.sol
> Compiling ./contracts/PumaPayPullPaymentV2.sol
> Compiling ./contracts/ownership/PayableOwnable.sol
> Compiling openzeppelin-solidity/contracts/access/Roles.sol
> Compiling openzeppelin-solidity/contracts/access/roles/Min
terRole.sol
> Compiling openzeppelin-solidity/contracts/math/SafeMath.so
l
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC2
0.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/ERC2
0Mintable.sol
> Compiling openzeppelin-solidity/contracts/token/ERC20/IERC
20.sol
> Artifacts written to /home/pavelcore/projects/smartdec/pum
```

```
apay/june/smart-contracts/build/contracts
> Compiled successfully using:
   - solc: 0.5.8+commit.23d335f2.Emscripten.clang
```

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 addres
s specified on contract deployment
       PumaPay Pull Payment deployment should revert when th
e token is a ZERO address
    Add executor
       should set the executor specified to true
       should transfer ETHER to the executor account for pay
ing 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 "LogSetConversionRate" event
    Register Pull Payment
       should add the pull payment for the beneficiary in th
e 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 th
e paymentExecutorOne to NOW
```

```
       should revert when NOT executed by an executor
       should revert when the payment for the beneficiary do
es not exists
       should revert when the deletion pull payment params d
oes 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 detai
ls to the paymentExecutorOne
       should update the pull payment numberOfPayments
       should update the pull payment nextPaymentTimestamp
       should update the pull payment lastPaymentTimestamp
       should revert if NOT executed by the executor
       should revert if executed before the start date speci
fied in the payment
       should revert when executed twice, i.e. number of pay
ments is zero
       should revert when pull payment does not exists for b
eneficiary calling the smart contract
       should emit a "LogPullPaymentExecuted" event
    Execute Recurring Pull Payment
       should pull the amount specified on the payment detai
ls 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 dat
e is reached
       should revert when if the next payment date is NOT re
ached
       should allow the merchant to pull payments in case th
ey have missed few payments
       should allow the merchant to pull payments in case th
ey have missed few payments and the customer cancelled the s
ubscription
    Execute Recurring Pull Payment with initial amount
       should pull the initial amount specified on the payme
nt 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 specifie
d for the reccuring payment to the beneficiary
       should set the intial payment amount to ZERO after pu
```

```
lling 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 and register a pull payment
    Delete Pull Payment
       should transfer ETH to the executor when its balance
is lower than 0.01 ETH

  Contract: PumaPay Pull Payment V2 Contract
    Deploying
       PumaPay Pull Payment owner should be the address that
was specified on contract deployment
       PumaPay Pull Payment token should be the token addres
s specified on contract deployment
       PumaPay Pull Payment deployment should revert when th
e token is a ZERO address
    Add executor
       should set the executor specified to true
       should transfer ETHER to the executor account for pay
ing gas fees
       should revert when the executor is a ZERO address
       should revert when 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
    Register Single Pull Payment
       should add the pull payment for the beneficiary in th
e active payments array
       should execute the single pull payment
       should revert when NOT executed by an executor
```

```
       should revert when the pull payment params does match
with the ones signed by the signatory
       should emit a "LogPaymentRegistered" event
       should emit a "LogPullPaymentExecution" event
    Register Recurring Pull Payment
       should add the pull payment for the beneficiary in th
e active payments array
       should execute the first payment from the recurring p
ull payment
       should execute payments from the recurring pull payme
nt: first payment and next one
       should execute payments from the recurring pull payme
nt: first payment, next one with third execution failing
       should revert when NOT executed by an executor
       should revert when the pull payment params does match
with the ones signed by the signatory
       should emit a "LogPaymentRegistered" event
       should emit a "LogPullPaymentExecution" event
    Register Recurring Pull Payment with Initial Payment
       should add the pull payment for the beneficiary in th
e active payments array
       should execute the first payment from the recurring p
ull payment
       should execute payments from the recurring pull payme
nt: first payment and next one
       should execute payments from the recurring pull paym
ent: initial payment, next two payments with third execution
failing
       should revert when NOT executed by an executor
       should revert when the pull payment params does match
with the ones signed by the signatory
       should emit a "LogPaymentRegistered" event
       should emit a "LogPullPaymentExecution" event
    Register Recurring Pull Payment with Free Trial
       should add the pull payment for the beneficiary in th
e active payments array
       should execute payments from the recurring pull payme
nt after the free trial has passed
       should execute payments from the recurring pull payme
nt after free trial: two payments with third execution faili
ng
       should revert if the free trial has not passed
       should revert when NOT executed by an executor
       should revert when the pull payment params does match
```

```
with the ones signed by the signatory
       should emit a "LogPaymentRegistered" event
    Register Recurring Pull Payment with Paid Trial
       should add the pull payment for the beneficiary in th
e active payments array
       should execute the initial payment for the paid trial
       should execute payments from the recurring pull payme
nt with paid trial: intial payment, two recurring payments w
ith third execution failing
       pull payment execution for recurring payment should r
evert if the paid trial has not passed
       should revert when NOT executed by an executor
       should revert when the pull payment params does match
with the ones signed by the signatory
       should emit a "LogPaymentRegistered" event
       should emit a "LogPullPaymentExecution" event
    Cancel Recurring Pull Payment
       should set the cancel date of the pull payment for th
e paymentExecutorOne to NOW
       should revert when NOT executed by an executor
       should revert when the payment for the beneficiary do
es not exists
       should revert when the deletion pull payment params d
oes match with the ones signed by the signatory
       should emit a "LogPaymentCancelled" event
    Execute pull payment
       should execute successfully a recurring pull payment
       should update the pull payment with next payment time
stamp
       should update the pull payment with last payment time
stamp
       should update the pull payment with number of payment
s
       should allow for pull payment executions to happen ev
en though some executions were missed
       should allow for pull payment executions to happen ev
en though some executions were missed and the customer has c
ancelled the subscription
       should fail when next payment timestamp has not passe
d
       should fail when pull payment was cancelled
       should emit a "LogPullPaymentExecution" event
    Add Executor - Funding
       should transfer ETH to the owner when its balance is
lower than 0.01 ETH
```

```
    Remove Executor - Funding
       should transfer ETH to the owner when its balance is
lower than 0.01 ETH
    Register Pull Payment - Funding
       should transfer ETH to the executor when its balance
is lower than 0.01 ETH and register a pull payment
    Delete Pull Payment - Funding
       should transfer ETH to the executor when its balance
is lower than 0.01 ETH


  119 passing (1m)
```

Solhint output

```
contracts/Migrations.sol
   5:17  error  Variable name must be in mixedCase        va
r-name-mixedcase
  19:30  error  Function param name must be in mixedCase  fu
nc-param-name-mixedcase

contracts/PumaPayPullPayment.sol
   45:2   error    Line length must be no more than 120 but
current length is 129          max-line-length
   46:2   error    Line length must be no more than 120 but
current length is 128          max-line-length
   49:2   error    Line length must be no more than 120 but
current length is 133          max-line-length
  110:10  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  111:9   warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  118:2   error    Line length must be no more than 120 but
current length is 140          max-line-length
  163:33  warning  Code contains empty block
o-empty-blocks
  182:9   warning  Possible reentrancy vulnerabilities. Avoi
d state changes after transfer  reentrancy
  232:2   error    Line length must be no more than 120 but
current length is 122          max-line-length
  274:2   error    Line length must be no more than 120 but
current length is 127          max-line-length
```

```
  337:2   error    Line length must be no more than 120 but
current length is 127          max-line-length
  339:73  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  394:2   error    Line length must be no more than 120 but
current length is 124          max-line-length
  397:68  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  420:2   error    Line length must be no more than 120 but
current length is 121          max-line-length
  439:2   error    Line length must be no more than 120 but
current length is 124          max-line-length

contracts/PumaPayPullPaymentV2.sol
  125:10  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  126:9   warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  189:33  warning  Code contains empty block
o-empty-blocks
  207:9   warning  Possible reentrancy vulnerabilities. Avoi
d state changes after transfer  reentrancy
  238:2   error    Line length must be no more than 120 but
current length is 122          max-line-length
  256:2   error    Line length must be no more than 120 but
current length is 124          max-line-length
  263:5   error    Function body contains 98 lines but allow
ed no more than 50 lines       function-max-lines
  278:2   error    Line length must be no more than 120 but
current length is 125          max-line-length
  279:2   error    Line length must be no more than 120 but
current length is 124          max-line-length
  280:2   error    Line length must be no more than 120 but
current length is 132          max-line-length
  281:2   error    Line length must be no more than 120 but
current length is 122          max-line-length
  282:2   error    Line length must be no more than 120 but
current length is 131          max-line-length
  283:2   error    Line length must be no more than 120 but
current length is 123          max-line-length
  284:2   error    Line length must be no more than 120 but
current length is 125          max-line-length
  320:2   error    Line length must be no more than 120 but
current length is 124          max-line-length
```

```
  325:2   error    Line length must be no more than 120 but
current length is 127          max-line-length
  329:79  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  331:2   error    Line length must be no more than 120 but
current length is 124          max-line-length
  335:2   error    Line length must be no more than 120 but
current length is 123          max-line-length
  339:79  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  341:2   error    Line length must be no more than 120 but
current length is 124          max-line-length
  345:2   error    Line length must be no more than 120 but
current length is 123          max-line-length
  350:79  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  352:2   error    Line length must be no more than 120 but
current length is 124          max-line-length
  391:2   error    Line length must be no more than 120 but
current length is 134          max-line-length
  393:80  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  440:2   error    Line length must be no more than 120 but
current length is 129          max-line-length
  443:2   error    Line length must be no more than 120 but
current length is 125          max-line-length
  445:2   error    Line length must be no more than 120 but
current length is 132          max-line-length
  446:74  warning  Avoid to make time-based decisions in you
r business logic               not-rely-on-time
  492:2   error    Line length must be no more than 120 but
current length is 121          max-line-length

 49 problems (34 errors, 15 warnings)
```

Solium output

```
contracts/PumaPayPullPayment.sol
  110:9     warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                  s
ecurity/no-block-members
  111:8     warning    Avoid using 'now' (alias to 'block.ti
```

```
mestamp').                                                    s
ecurity/no-block-members
  114:16    error      Only use indent of 12 spaces.
ndentation
  117:16    error      Only use indent of 12 spaces.
ndentation
  120:16    error      Only use indent of 12 spaces.
ndentation
  289:16    error      Only use indent of 12 spaces.
ndentation
  290:16    error      Only use indent of 12 spaces.
ndentation
  291:16    error      Only use indent of 12 spaces.
ndentation
  292:16    error      Only use indent of 12 spaces.
ndentation
  293:16    error      Only use indent of 12 spaces.
ndentation
  294:16    error      Only use indent of 12 spaces.
ndentation
  295:16    error      Only use indent of 12 spaces.
ndentation
  339:72    warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  392:12    warning    Assignment operator must have exactly
single space on both sides of it.                           o
perator-whitespace
  397:67    warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  503:35    warning    Operator "&&" should be on the line w
here left side of the Binary expression ends.               o
perator-whitespace
  512:23    error      Only use indent of 12 spaces.
ndentation
  512:51    warning    'keccak256': The last argument must n
ot be succeeded by any whitespace or comments (only ')').    f
unction-whitespace

contracts/PumaPayPullPaymentV2.sol
  124:16    error      Only use indent of 12 spaces.
ndentation
  125:9     warning    Avoid using 'now' (alias to 'block.ti
```

```
mestamp').                                                    s
ecurity/no-block-members
  126:8     warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  129:16    error      Only use indent of 12 spaces.
ndentation
  137:16    error      Only use indent of 12 spaces.
ndentation
  304:16    error      Only use indent of 12 spaces.
ndentation
  305:16    error      Only use indent of 12 spaces.
ndentation
  306:16    error      Only use indent of 12 spaces.
ndentation
  307:16    error      Only use indent of 12 spaces.
ndentation
  308:16    error      Only use indent of 12 spaces.
ndentation
  309:16    error      Only use indent of 12 spaces.
ndentation
  310:16    error      Only use indent of 12 spaces.
ndentation
  329:78    warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  339:78    warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  350:78    warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  393:79    warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  442:8     warning    Assignment operator must have exactly
single space on both sides of it.                           o
perator-whitespace
  446:73    warning    Avoid using 'now' (alias to 'block.ti
mestamp').                                                    s
ecurity/no-block-members
  575:42    warning    Operator "&&" should be on the line w
here left side of the Binary expression ends.               o
perator-whitespace
```

```
  584:23    error      Only use indent of 12 spaces.
ndentation
  584:51    warning    'keccak256': The last argument must n
ot be succeeded by any whitespace or comments (only ')').    f
unction-whitespace

contracts/ownership/PayableOwnable.sol
  38:8    warning    Provide an error message for require().
error-reason
  73:8    warning    Provide an error message for require().
error-reason

 22 errors, 19 warnings found.
```