smartypay

SMART CONTRACT AUDIT

Contents

Document properties

Version	Date	Author	Description
0.1	16.03.2023	Dmitry Sapaev	Initial Draft

Contacts:
Dmitry Sapaev
d.sapaev@smartypay.io

Introduction

SMARTy Pay is a multi-chain, non-custodial crypto payment platform with different billing methods allowing to accept cryptocurrency payments to merchants from customers around the world.

The SMARTy payment protocol is fully decentralized, enabling maximum control over the merchant's assets. The solution is designed in a way where almost every step of payment flow is controlled by smart contracts logic located on the blockchain.

Given that there is no central trusted party between a customer and a merchant, that can fail or somehow block the flow of funds.

Service can operate almost autonomously if the blockchain network works correctly.

This document results from the SMARTy Pay smart contracts audit, highlighting potential security issues, recommendations for optimization, and improvements.

Overall, 187 vulnerabilities were found and fixed due to the audit.

Audited smart contracts list

Here is the list of SMARTy Pay smart contracts. Some of these contracts are abstract or libraries and do not have deployment addresses.

№	Contract name	Deployed address*
1	BaseMinimalProxyFactory	-
2	MerchantWalletFactory	0x5977163FBc9b056bDc5573f0A557a0367C76E92c
3	BasePaymentFactory	-
4	PullPaymentFactory	0xc9ec071B3315d4a1cC0Fa906e3250480D479B0c2
5	PushPaymentFactory	0xEE0668dB96C4eDcB2794Fe0c7f16A264dBE3cb84
6	SimplePullPaymentFactory	0x297D0E7542893DAdfb08Cd5feCD61559432fE2CB
7	SimplePushPaymentFactory	0x385b5Bb0F351C52bcfBf3fe7AD64F00590D04858
8	VestingWalletFactory	0x3cF0965b063FbE7fa85449C0Df772F00dd23957e
9	WalletFactory	0xdA48BA9A84CEd0FC284E4D762D8a8AF5E62477F5
10	KYC	-
11	StubBlackListChecker	0x36354cC644eb0C2CA6fBCBd11a57AB1aB02aefdB
12	StubKYC	-
13	BaseSimplePayment	-
14	SimplePullPayment	0x1bADa81b8B3437d83242f40f34941dE136c98583
15	SimplePushPayment	-
16	BasePayment	-
17	BasePullPayment	-
18	PullPayment	0xaA8C186b8BD134e97336D7d7E812aBF462abFd66
19	PushPayment	0xaf89d8E6403a643aC8b60fad57c3319B554056AD
20	SmartyProcessingPayments	0xB72aAd7e871BB7435168e685e2CD6C37B6bfC2cc
21	BaseFinanceRouter	-
22	UniswapV2Router	0x966B6D126E3dCABD52a20cccD09FAbC50622Ca59
23	SmartyProcessingWithdrawals	0x58fb8Dfe009d422e78d90A47430133C2446668D8

№	Contract name	Deployed address*
24	ProcessingEvents	-
25	CustomerWallet	-
26	BSCCustomerWallet	0xb366F89dBdc799FB98EEAc24aF86897a862c1865
27	MerchantWallet	0x5Af94aE4E41B24C137d20AC15493630C9E67A509
28	VestingWallet	0x4E32683Cb860C57aDD09b6eD54C00Ede2dB05003
29	BaseFarm	-
30	LiquidityFarm	-
31	RewardsFarm	-
32	BaseRewardsController	-
33	BSCRewardsController	-
34	MultichainRewardsController	-
35	Migrations	-
36	SafeTransfer	-
37	SmartyPublisher	0xa93CC92Fec72E1d0604545608322b01D630Daf4d
38	Notifier	0xE9479C434e4AE232CdE5c97dDb2072A63b499D8c
39	ContractsRegistry	0xB7911236694040A7B4B2e41fA6B22E87c8BE0a1B
40	BatchOfBalances	-
41	SmartyCryptoProcessing	0x149622EF78d2DE74e9B14fDA5da15fda6918A06B
42	SmartyCryptoProcessingDelegate	0x8659Bf731822796Bc5C793Be598A58CF7B84Ffb5
43	SmartyProcessingManager	0x8C785dB8fA7E67725BEC59b442Ae7B35f0ce70E5
44	Ownable	-
45	PullPaymentFactoryLib	-
46	SafeMath	-
47	SmartyTransfer	-
48	Structs	-
49	UniswapV2Library	-

* All addresses are the same for Binance, Polygon, Arbitrum and Fantom testnets and mainnets.

Audit methodology

For the initial audit of smart contracts, we used a set of different methodologies:

- **Automated weaknesses search.**

We used the Slither tool (Solidity static analyzer) for automated weaknesses search and integrated it into our CI.

- **Codel logic analysis.**

We have analyzed all functions for their efficiency and correctness.

- **Access control.**

We have analyzed all our contracts for the possibility of unauthorized access and withdrawal of funds

Vulnerability classification

Here's the list of all known vulnerabilities that were used to analyse smart contracts.

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
1	CWS-100		Functions that do not have a function visibility type specified are public by default. This can lead to a vulnerability if a developer forgot to set the visibility and a malicious user is able to make unauthorized or unintended state changes.	
2	CWS-101		An overflow/underflow happens when an arithmetic operation reaches the maximum or minimum size of a type. For instance if a number is stored in the uint8 type, it means that the number is stored in a 8 bits unsigned number ranging from 0 to 2^8-1. In computer programming, an integer overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits – either larger than the maximum or lower than the minimum representable value.	
3	CWS-102		Using an outdated compiler version can be problematic especially if there are publicly disclosed bugs and issues that affect the current compiler version.	
4	CWS-103	solc-version	Contracts should be deployed with the same compiler version and flags that they have been tested with thoroughly. Locking the pragma helps to ensure that contracts do not accidentally get deployed using, for example, an outdated compiler version that might introduce bugs that affect the contract system negatively.	Informational
5	CWS-104	unchecked-lowlevel	The return value of a message call is not checked. Execution will resume even if the called contract throws an exception. If the call fails accidentally or an attacker forces the call to fail, this may cause unexpected behaviour in the subsequent program logic.	Medium
6	CWS-105		Due to missing or insufficient access controls, malicious parties can withdraw some or all Ether from the contract account.This bug is sometimes caused by unintentionally exposing initialization functions. By wrongly naming a function intended to be a constructor, the constructor code ends up in the runtime byte code and can be called by anyone to re-initialize the contract.	
7	CWS-106	suicidal	Due to missing or insufficient access controls, malicious parties can self-destruct the contract.	High
8	CWS-107	reentrancy-eth	One of the major dangers of calling external contracts is that they can take over the control flow. In the reentrancy attack (a.k.a. recursive call attack), a malicious contract calls back into the calling contract before the first invocation of the function is finished. This may cause the different invocations of the function to interact in undesirable ways.	High

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
9	CWS-108		Labeling the visibility explicitly makes it easier to catch incorrect assumptions about who can access the variable.	
10	CWS-109	uninitialized-local	Uninitialized local storage variables can point to unexpected storage locations in the contract, which can lead to intentional or unintentional vulnerabilities.	Medium
11	CWS-110		The Solidity assert() function is meant to assert invariants. Properly functioning code should never reach a failing assert statement. A reachable assertion can mean one of two things:A bug exists in the contract that allows it to enter an invalid state;The assert statement is used incorrectly, e.g. to validate inputs.	
12	CWS-111		Several functions and operators in Solidity are deprecated. Using them leads to reduced code quality. With new major versions of the Solidity compiler, deprecated functions and operators may result in side effects and compile errors.	
13	CWS-112	controlled-delegatecall	There exists a special variant of a message call, named delegatecall which is identical to a message call apart from the fact that the code at the target address is executed in the context of the calling contract and msg.sender and msg.value do not change their values. This allows a smart contract to dynamically load code from a different address at runtime. Storage, current address and balance still refer to the calling contract.Calling into untrusted contracts is very dangerous, as the code at the target address can change any storage values of the caller and has full control over the caller's balance.	High
14	CWS-113	calls-loop	External calls can fail accidentally or deliberately, which can cause a DoS condition in the contract. To minimize the damage caused by such failures, it is better to isolate each external call into its own transaction that can be initiated by the recipient of the call. This is especially relevant for payments, where it is better to let users withdraw funds rather than push funds to them automatically (this also reduces the chance of problems with the gas limit).	Low
15	CWS-114		Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	
16	CWS-115	tx-origin	tx.origin is a global variable in Solidity which returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable if an authorized account calls into a malicious contract. A call could be made to the vulnerable contract that passes the authorization check since tx.origin returns the original sender of the transaction which in this case is the authorized account.	Medium

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
17	CWS-116	timestamp	Block values as a proxy for time	Low
18	CWS-117		The implementation of a cryptographic signature system in Ethereum contracts often assumes that the signature is unique, but signatures can be altered without the possession of the private key and still be valid. The EVM specification defines several so-called 'precompiled' contracts one of them being ecrecover which executes the elliptic curve public key recovery. A malicious user can slightly modify the three values v, r and s to create other valid signatures. A system that performs signature verification on contract level might be susceptible to attacks if the signature is part of the signed message hash. Valid signatures could be created by a malicious user to replay previously signed messages.	
19	CWS-118		Constructors are special functions that are called only once during the contract creation. They often perform critical, privileged actions such as setting the owner of the contract. Before Solidity version 0.4.22, the only way of defining a constructor was to create a function with the same name as the contract class containing it. A function meant to become a constructor becomes a normal, callable function if its name doesn't exactly match the contract name. This behavior sometimes leads to security issues, in particular when smart contract code is re-used with a different name but the name of the constructor function is not changed accordingly.	
20	CWS-119		Solidity allows for ambiguous naming of state variables when inheritance is used. Contract A with a variable x could inherit contract B that also has a state variable x defined. This would result in two separate versions of x, one of them being accessed from contract A and the other one from contract B. In more complex contract systems this condition could go unnoticed and subsequently lead to security issues.Shadowing state variables can also occur within a single contract when there are multiple definitions on the contract and function level.	
21	CWS-120	weak-prng	Ability to generate random numbers is very helpful in all kinds of applications. One obvious example is gambling DApps, where pseudo-random number generator is used to pick the winner. However, creating a strong enough source of randomness in Ethereum is very challenging. For example, use of block.timestamp is insecure, as a miner can choose to provide any timestamp within a few seconds and still get his block accepted by others. Use of blockhash, block.difficulty and other fields is also insecure, as they're controlled by the miner. If the stakes are high, the miner can mine lots of blocks in a short time by renting hardware, pick the block that has required block hash for him to win, and drop all others.	High

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
22	CWS-121		It is sometimes necessary to perform signature verification in smart contracts to achieve better usability or to save gas cost. A secure implementation needs to protect against Signature Replay Attacks by for example keeping track of all processed message hashes and only allowing new message hashes to be processed. A malicious user could attack a contract without such a control and get message hash that was sent by another user processed multiple times.	
23	CWS-122		It is a common pattern for smart contract systems to allow users to sign messages off-chain instead of directly requesting users to do an on-chain transaction because of the flexibility and increased transferability that this provides. Smart contract systems that process signed messages have to implement their own logic to recover the authenticity from the signed messages before they process them further. A limitation for such systems is that smart contracts can not directly interact with them because they can not sign messages. Some signature verification implementations attempt to solve this problem by assuming the validity of a signed message based on other methods that do not have this limitation. An example of such a method is to rely on msg.sender and assume that if a signed message originated from the sender address then it has also been created by the sender address. This can lead to vulnerabilities especially in scenarios where proxies can be used to relay transactions.	
24	CWS-123		The Solidity require() construct is meant to validate external inputs of a function. In most cases, such external inputs are provided by callers, but they may also be returned by callees. In the former case, we refer to them as precondition violations. Violations of a requirement can indicate one of two possible issues:A bug exists in the contract that provided the external input.The condition used to express the requirement is too strong.	
25	CWS-124		A smart contract's data (e.g., storing the owner of the contract) is persistently stored at some storage location (i.e., a key or address) on the EVM level. The contract is responsible for ensuring that only authorized user or contract accounts may write to sensitive storage locations. If an attacker is able to write to arbitrary storage locations of a contract, the authorization checks may easily be circumvented. This can allow an attacker to corrupt the storage; for instance, by overwriting a field that stores the address of the contract owner.	

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
26	CWS-125		Solidity supports multiple inheritance, meaning that one contract can inherit several contracts. Multiple inheritance introduces ambiguity called Diamond Problem: if two or more base contracts define the same function, which one should be called in the child contract? Solidity deals with this ambiguity by using reverse C3 Linearization, which sets a priority between base contracts.That way, base contracts have different priorities, so the order of inheritance matters. Neglecting inheritance order can lead to unexpected behavior.	
27	CWS-126		Insufficient gas griefing attacks can be performed on contracts which accept data and use it in a sub-call on another contract. If the sub-call fails, either the whole transaction is reverted, or execution is continued. In the case of a relayer contract, the user who executes the transaction, the 'forwarder', can effectively censor transactions by using just enough gas to execute the transaction, but not enough for the sub-call to succeed.	
28	CWS-127	assembly	Solidity supports function types. That is, a variable of function type can be assigned with a reference to a function with a matching signature. The function saved to such variable can be called just like a regular function.The problem arises when a user has the ability to arbitrarily change the function type variable and thus execute random code instructions. As Solidity doesn't support pointer arithmetics, it's impossible to change such variable to an arbitrary value. However, if the developer uses assembly instructions, such as mstore or assign operator, in the worst case scenario an attacker is able to point a function type variable to any code instruction, violating required validations and required state changes.	Informational
29	CWS-128		When smart contracts are deployed or functions inside them are called, the execution of these actions always requires a certain amount of gas, based of how much computation is needed to complete them. The Ethereum network specifies a block gas limit and the sum of all transactions included in a block can not exceed the threshold.Programming patterns that are harmless in centralized applications can lead to Denial of Service conditions in smart contracts when the cost of executing a function exceeds the block gas limit. Modifying an array of unknown size, that increases in size over time, can lead to such a Denial of Service condition.	
30	CWS-129		A typographical error can occur for example when the intent of a defined operation is to sum a number to a variable (+=) but it has accidentally been defined in a wrong way (=+), introducing a typo which happens to be a valid operator. Instead of calculating the sum it initializes the variable again.The unary + operator is deprecated in new solidity compiler versions.	

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
31	CWS-130		Malicious actors can use the Right-To-Left-Override unicode character to force RTL text rendering and confuse users as to the real intent of a contract.	
32	CWS-131	unused-state	Unused variables are allowed in Solidity and they do not pose a direct security issue. It is best practice though to avoid them as they can:cause an increase in computations (and unnecessary gas consumption)indicate bugs or malformed data structures and they are generally a sign of poor code qualitycause code noise and decrease readability of the code	Informational
33	CWS-132	incorrect-equality	Contracts can behave erroneously when they strictly assume a specific Ether balance. It is always possible to forcibly send ether to a contract (without triggering its fallback function), using selfdestruct, or by mining to the account. In the worst case scenario this could lead to DOS conditions that might render the contract unusable.	Medium
34	CWS-133		Using abi.encodePacked() with multiple variable length arguments can, in certain situations, lead to a hash collision. Since abi.encodePacked() packs all elements in order regardless of whether they're part of an array, you can move elements between arrays and, so long as all elements are in the same order, it will return the same encoding. In a signature verification situation, an attacker could exploit this by modifying the position of elements in a previous function call to effectively bypass authorization	
35	CWS-134	reentrancy-unlimited-gas	The transfer() and send() functions forward a fixed amount of 2300 gas. Historically, it has often been recommended to use these functions for value transfers to guard against reentrancy attacks. However, the gas cost of EVM instructions may change significantly during hard forks which may break already deployed contract systems that make fixed assumptions about gas costs. For example. EIP 1884 broke several existing smart contracts due to a cost increase of the SLOAD instruction.	Medium
36	CWS-135	dead-code	In Solidity, it's possible to write code that does not produce the intended effects. Currently, the solidity compiler will not return a warning for effect-free code. This can lead to the introduction of "dead" code that does not properly performing an intended action.For example, it's easy to miss the trailing parentheses in msg.sender.call.value(address(this).balance)("");, which could lead to a function proceeding without transferring funds to msg.sender. Although, this should be avoided by checking the return value of the call.	Informational

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
37	CWS-136		It is a common misconception that private type variables cannot be read. Even if your contract is not published, attackers can look at contract transactions to determine values stored in the state of the contract. For this reason, it's important that unencrypted private data is not stored in the contract code or state.	
38	-	arbitrary-send-eth	Unprotected call to a function sending Ether to an arbitrary address.	High
39	-	name-reused	If a codebase has two contracts the similar names, the compilation artifacts will not contain one of the contracts with the duplicate name.	High
40	-	unchecked-transfer	The return value of an external transfer/transferFrom call is not checkedSeveral tokens do not revert in case of failure and return false. If one of these tokens is used in MyBank, deposit will not revert if the transfer fails, and an attacker can call deposit for free..	High
41	-	uninitialized-state	Uninitialized state variables	High
42	-	divide-before-multiply	Solidity's integer division truncates. Thus, performing division before multiplication can lead to precision loss.	Medium
43	-	locked-ether	Contract with a payable function, but without a withdrawal capacity. Every Ether sent to Locked will be lost.	Medium
44	-	reentrancy-no-eth	Detection of the reentrancy bug. Do not report reentrancies that involve Ether (see reentrancy-eth).	Medium
45	-	tautology	Detects expressions that are tautologies or contradictions.	Medium
46	-	unchecked-send	The return value of a send is not checked. The return value of send is not checked, so if the send fails, the Ether will be locked in the contract. If send is used to prevent blocking operations, consider logging the failed send.	Medium
47	-	unused-return	The return value of an external call is not stored in a local or state variable.	Medium
48	-	shadowing-builtin	Detection of shadowing built-in symbols using local variables, state variables, functions, modifiers, or events.	Low
49	-	shadowing-local	Detection of shadowing using local variables.	Low
50	-	events-access	Detect missing events for critical access control parameters	Low
51	-	events-maths	Detect missing events for critical arithmetic parameters	Low
52	-	missing-zero-check	Detect missing zero address validation	Low

№	Smart Contract Weakness Classification	Weakness code	Brief description	Severity
53	-	incorrect-modifier	If a modifier does not execute _ or revert, the execution of the function will return the default value, which can be misleading for the caller	Low
54	-	variable-scope	Detects the possible usage of a variable before the declaration is stepped over (either because it is later declared, or declared in another scope)	Low
55	-	reentrancy-benign	Detection of the reentrancy bug. Only report reentrancy that acts as a double call (see reentrancy-eth, reentrancy-no-eth)	Low
56	-	reentrancy-events	Detection of the reentrancy bug. Only report reentrancies leading to out-of-order events	Low
57	-	boolean-equal	Detects the comparison to boolean constants	Informational
58	-	costly-loop	Costly operations inside a loop might waste gas, so optimizations are justified	Informational
59	-	function-init-state	Detects the immediate initialization of state variables through function calls that are not pure/constant, or that use non-constant state variable	Informational
60	-	low-level-calls	The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success	Informational
61	-	missing-inheritance	Detect missing inheritance	Informational
62	-	naming-convention	Solidity defines a naming convention that should be followed	Informational
63	-	redundant-statements	Detect the usage of redundant statements that have no effect	Informational
64	-	similar-names	Detect variables with names that are too similar	Informational
65	-	too-many-digits	Literals with many digits are difficult to read and review	Informational
66	-	constable-states	State variables that are not updated following deployment should be declared constant to save gas	Optimization
67	-	arbitrary-send-erc20	Detects when msg.sender is not used as from in transferFrom.	High

Detailed results

arbitrary-send-erc20

Arbitrary `from` in transferFrom

Configuration

- Check: `arbitrary-send-erc20`
- Severity: `High`
- Confidence: `High`

Description

Detect when `msg.sender` is not used as `from` in transferFrom.

Exploit Scenario:

```
function a(address from, address to, uint256 amount) public {
    erc20.transferFrom(from, to, am);
}
```

Alice approves this contract to spend her ERC20 tokens. Bob can call `a` and specify Alice's address as the `from` parameter in `transferFrom`, allowing him to transfer Alice's tokens to himself.

Recommendation

Use `msg.sender` as `from` in transferFrom.

- ID-0

[PullPayment.providePayment()](contracts/solidity/payments/PullPayment.sol#L35-L38) uses arbitrary from in transferFrom: [require(bool,string)(IERC20(token).transferFrom(payer,address(this),amount),NCPS-34.021:TRANSFER_FAILED)](contracts/solidity/payments/PullPayment.sol#L36)

contracts/solidity/payments/PullPayment.sol#L35-L38

- ID-1

[SafeTransfer.forward(address,bytes32,uint256,address)](contracts/solidity/onramp/SafeTransfer.sol#L20-L25) uses arbitrary from in transferFrom: [IERC20(token).transferFrom(transferWallet,recipient,amount)](contracts/solidity/onramp/SafeTransfer.sol#L24)

contracts/solidity/onramp/SafeTransfer.sol#L20-L25

- ID-2

[SimplePullPayment.providePayment()](contracts/solidity/payments/simple/SimplePullPayment.sol#L25-L29) uses arbitrary from in transferFrom: [require(bool,string)(IERC20(token).transferFrom(payer,merchantWallet,amount),NCPS-34.021:TRANSFER_FAILED)](contracts/solidity/payments/simple/SimplePullPayment.sol#L27)

contracts/solidity/payments/simple/SimplePullPayment.sol#L25-L29

- ID-3

[BaseFinanceRouter.pay(address,address,address,uint256)](contracts/solidity/BaseFinanceRouter.sol#L17-L20) uses arbitrary from in transferFrom: [require(bool,string)(IERC20(token).transferFrom(from,to,value),NCPS-03.005:TRANSFER_FAILED)](contracts/solidity/BaseFinanceRouter.sol#L19)

contracts/solidity/BaseFinanceRouter.sol#L17-L20

arbitrary-send-eth

Functions that send Ether to arbitrary destinations

Configuration

- Check: `arbitrary-send-eth`
- Severity: `High`
- Confidence: `Medium`

Description

Unprotected call to a function sending Ether to an arbitrary address.

Exploit Scenario:

```
contract ArbitrarySendEth{
    address destination;
    function setDestination(){
        destination = msg.sender;
    }

    function withdraw() public{
        destination.transfer(this.balance);
    }
}
```

Bob calls `setDestination` and `withdraw`. As a result he withdraws the contract's balance.

Recommendation

Ensure that an arbitrary user cannot withdraw unauthorized funds.

- ID-4
[BaseSimplePayment.flush(address)](contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38) sends eth to arbitrary user
 Dangerous calls:
 - [require(bool,string)(address(merchantWallet).send(address(this).balance),NCPS-38.021:TRANSFER_FAILED)]
(contracts/solidity/payments/simple/BaseSimplePayment.sol#L36)

contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38

- ID-5
[PushPayment.withdrawEth(address,uint256,uint8,bytes32,bytes32,address)]
(contracts/solidity/payments/PushPayment.sol#L19-L29) sends eth to arbitrary user
 Dangerous calls:
 - [IProcessingPayments(paymentsEngine).pay{value: amount}(stealth,type()(uint256).max,path,type()(uint256).max)]
(contracts/solidity/payments/PushPayment.sol#L27)

contracts/solidity/payments/PushPayment.sol#L19-L29

- ID-6
[VestingWallet._burnLiquidityFarmToken(uint256)](contracts/solidity/vesting/VestingWallet.sol#L88-L95) sends eth to arbitrary user
 Dangerous calls:
 - [investor.transfer(address(this).balance)](contracts/solidity/vesting/VestingWallet.sol#L94)

contracts/solidity/vesting/VestingWallet.sol#L88-L95

- ID-7
[SimplePushPayment.withdrawEth(uint256)](contracts/solidity/payments/simple/SimplePushPayment.sol#L28-L34) sends eth to arbitrary user
 Dangerous calls:
 - [require(bool,string)(address(merchantWallet).send(address(this).balance),NCPS-39.012:TRANSFER_FAILED)] (contracts/solidity/payments/simple/SimplePushPayment.sol#L33)

contracts/solidity/payments/simple/SimplePushPayment.sol#L28-L34

reentrancy-eth

Reentrancy vulnerabilities

Configuration

- Check: `reentrancy-eth`
- Severity: `High`
- Confidence: `Medium`

Description

Detection of the reentrancy bug. Do not report reentrancies that don't involve Ether (see `reentrancy-no-eth`)

Exploit Scenario:

```
    function withdrawBalance(){
        // send userBalance[msg.sender] Ether to msg.sender
        // if msg.sender is a contract, it will call its fallback function
        if( ! (msg.sender.call.value(userBalance[msg.sender])() ) ){
            throw;
        }
        userBalance[msg.sender] = 0;
    }
```

Bob uses the re-entrancy bug to call `withdrawBalance` two times, and withdraw more than its initial deposit to the contract.

Recommendation

Apply the `check-effects-interactions pattern`.

- ID-8

Reentrancy in [PushPayment.withdrawEth(address,uint256,uint8,bytes32,bytes32,address)]
(contracts/solidity/payments/PushPayment.sol#L19-L29):
 External calls:
 - [path[0] = PaymentsAPI(paymentsEngine).router().WETH()](contracts/solidity/payments/PushPayment.sol#L26)
 - [IProcessingPayments(paymentsEngine).pay{value: amount}(stealth,type()(uint256).max,path,type()(uint256).max)]
(contracts/solidity/payments/PushPayment.sol#L27)
 External calls sending eth:
 - [IProcessingPayments(paymentsEngine).pay{value: amount}(stealth,type()(uint256).max,path,type()(uint256).max)]
(contracts/solidity/payments/PushPayment.sol#L27)
 State variables written after the call(s):
 - [stealth = newStealth](contracts/solidity/payments/PushPayment.sol#L28)
 [BasePayment.stealth](contracts/solidity/payments/BasePayment.sol#L9) can be used in cross function reentrancies:
 - [BasePayment.checkWithdrawalRequest(address,uint8,bytes32,bytes32,address)]
(contracts/solidity/payments/BasePayment.sol#L44-L51)
 - [PushPayment.fallback()](contracts/solidity/payments/PushPayment.sol#L31-L33)
 - [BasePayment.initStealth(address)](contracts/solidity/payments/BasePayment.sol#L13-L17)
 - [BasePayment.initToken(address)](contracts/solidity/payments/BasePayment.sol#L19-L22)
 - [BasePayment.stealth](contracts/solidity/payments/BasePayment.sol#L9)
 - [BasePayment.withdraw(address,uint256,uint8,bytes32,bytes32,address)](contracts/solidity/payments/BasePayment.sol#L24-L41)
 - [PushPayment.withdrawEth(address,uint256,uint8,bytes32,bytes32,address)]
(contracts/solidity/payments/PushPayment.sol#L19-L29)

contracts/solidity/payments/PushPayment.sol#L19-L29

suicidal

Suicidal

Configuration

- Check: `suicidal`
- Severity: `High`
- Confidence: `High`

Description

Unprotected call to a function executing `selfdestruct` / `suicide`.

Exploit Scenario:

```
contract Suicidal{
    function kill() public{
        selfdestruct(msg.sender);
    }
}
```

Bob calls `kill` and destructs the contract.

Recommendation

Protect access to all sensitive functions.

- ID-9
[PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61) allows anyone to destruct the contract

contracts/solidity/payments/PullPayment.sol#L40-L61

unchecked-transfer

Unchecked transfer

Configuration

- Check: `unchecked-transfer`
- Severity: `High`
- Confidence: `Medium`

Description

The return value of an external transfer/transferFrom call is not checked

Exploit Scenario:

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in `MyBank`, `deposit` will not revert if the transfer fails, and an attacker can call `deposit` for free..

Recommendation

Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked.

- ID-10
[LiquidityFarm.stakeToken(address,uint256,bool)](contracts/solidity/rewards/LiquidityFarm.sol#L24-L73) ignores return value by [IERC20(token0).transfer(stakedToken,exactAmountOut)](contracts/solidity/rewards/LiquidityFarm.sol#L58)

contracts/solidity/rewards/LiquidityFarm.sol#L24-L73

- ID-11
[LiquidityFarm.stakeToken(address,uint256,bool)](contracts/solidity/rewards/LiquidityFarm.sol#L24-L73) ignores return value by [IERC20(token).transferFrom(msg.sender,stakedToken,swapAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L49)

contracts/solidity/rewards/LiquidityFarm.sol#L24-L73

- ID-12
[SafeTransfer.forward(address,bytes32,uint256,address)](contracts/solidity/onramp/SafeTransfer.sol#L20-L25) ignores return value by [IERC20(token).transferFrom(transferWallet,recipient,amount)](contracts/solidity/onramp/SafeTransfer.sol#L24)

contracts/solidity/onramp/SafeTransfer.sol#L20-L25

- ID-13
[LiquidityFarm.stakeToken(address,uint256,bool)](contracts/solidity/rewards/LiquidityFarm.sol#L24-L73) ignores return value by [IERC20(token1).transfer(stakedToken,exactAmountOut)](contracts/solidity/rewards/LiquidityFarm.sol#L55)

contracts/solidity/rewards/LiquidityFarm.sol#L24-L73

- ID-14

[BaseRewardsController.distribute(address[],uint256)](contracts/solidity/rewards/BaseRewardsController.sol#L32-L39) ignores return value by [IERC20(rewardsToken).transfer(rewardsFarm,amount)](contracts/solidity/rewards/BaseRewardsController.sol#L36)

contracts/solidity/rewards/BaseRewardsController.sol#L32-L39

- ID-15

[SafeTransfer.flush(address,uint256)](contracts/solidity/onramp/SafeTransfer.sol#L33-L36) ignores return value by [IERC20(token).transfer(transferWallet,amount)](contracts/solidity/onramp/SafeTransfer.sol#L35)

contracts/solidity/onramp/SafeTransfer.sol#L33-L36

- ID-16

[LiquidityFarm.stakeToken(address,uint256,bool)](contracts/solidity/rewards/LiquidityFarm.sol#L24-L73) ignores return value by [IERC20(token).transferFrom(msg.sender,stakedToken,restAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L61)

contracts/solidity/rewards/LiquidityFarm.sol#L24-L73

incorrect-equality

Dangerous strict equalities

Configuration

- Check: `incorrect-equality`
- Severity: `Medium`
- Confidence: `High`

Description

Use of strict equalities that can be easily manipulated by an attacker.

Exploit Scenario:

```
contract Crowdsale{
    function fund_reached() public returns(bool){
        return this.balance == 100 ether;
    }
```

`Crowdsale` relies on `fund_reached` to know when to stop the sale of tokens. `Crowdsale` reaches 100 Ether. Bob sends 0.1 Ether. As a result, `fund_reached` is always false and the `crowdsale` never ends.

Recommendation

Don't use strict equality to determine if an account has enough Ether or tokens.

- ID-17
[SmartyProcessingPayments.internalPay(address,address[],uint256,int256,address,uint256,uint256,IFinanceRouter,bool)](contracts/solidity/payments/SmartyProcessingPayments.sol#L230-L249) uses a dangerous strict equality:
 - [require(bool,string)(amountOut > 0 || amountOut == - 1,NCPS-03.015:BAD_OUTPUT_AMOUNT)](contracts/solidity/payments/SmartyProcessingPayments.sol#L234)

contracts/solidity/payments/SmartyProcessingPayments.sol#L230-L249

locked-ether

Contracts that lock Ether

Configuration

- Check: `locked-ether`
- Severity: `Medium`
- Confidence: `High`

Description

Contract with a `payable` function, but without a withdrawal capacity.

Exploit Scenario:

```
pragma solidity 0.4.24;
contract Locked{
    function receive() payable public{
    }
}
```

Every Ether sent to `Locked` will be lost.

Recommendation

Remove the payable attribute or add a withdraw function.

- ID-18

Contract locking ether found:
 Contract [SmartyCryptoProcessingDelegate](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L18-L167) has payable functions:
  - [IProcessingPayments.pay(address,uint256,address[],uint256)](contracts/solidity/payments/interfaces/IProcessingPayments.sol#L21)
  - [SmartyCryptoProcessingDelegate.pay(address,uint256,address[],uint256)](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L68-L70)
 But does not have a function to withdraw the ether

contracts/solidity/SmartyCryptoProcessingDelegate.sol#L18-L167

reentrancy-no-eth

Reentrancy vulnerabilities

Configuration

- Check: `reentrancy-no-eth`
- Severity: `Medium`
- Confidence: `Medium`

Description

Detection of the reentrancy bug. Do not report reentrancies that involve Ether (see `reentrancy-eth`).

Exploit Scenario:

```
function bug(){
    require(not_called);
    if( ! (msg.sender.call() ) ){
        throw;
    }
    not_called = False;
}
```

Recommendation

Apply the `check-effects-interactions` pattern.

- ID-19

Reentrancy in [VestingWallet.updateLiquidityFarm(address)](contracts/solidity/vesting/VestingWallet.sol#L112-L116):
External calls:
- [IERC20(ILiquidityFarm(newFarm).stakedToken()).approve(newFarm,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L113)
- [_beforeUpdateFarm(liquidityFarm,newFarm)](contracts/solidity/vesting/VestingWallet.sol#L114)
- [(returnData) = oldFarm.call(payload)](contracts/solidity/vesting/VestingWallet.sol#L121)
- [(None,returnData) = oldFarm.call(payload)](contracts/solidity/vesting/VestingWallet.sol#L125)
- [IStakingFarm(newFarm).stake(stakedAmount)](contracts/solidity/vesting/VestingWallet.sol#L129)
State variables written after the call(s):
- [liquidityFarm = newFarm](contracts/solidity/vesting/VestingWallet.sol#L115)
[VestingWallet.liquidityFarm](contracts/solidity/vesting/VestingWallet.sol#L20) can be used in cross function reentrancies:
- [VestingWallet._burnLiquidityFarmToken(uint256)](contracts/solidity/vesting/VestingWallet.sol#L88-L95)
- [VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56)
- [VestingWallet.liquidityFarm](contracts/solidity/vesting/VestingWallet.sol#L20)
- [VestingWallet.stakeToLiquidityFarm(uint256)](contracts/solidity/vesting/VestingWallet.sol#L77-L82)
- [VestingWallet.updateLiquidityFarm(address)](contracts/solidity/vesting/VestingWallet.sol#L112-L116)
- [VestingWallet.withdrawFromLiquidityFarm(uint256)](contracts/solidity/vesting/VestingWallet.sol#L97-L101)

contracts/solidity/vesting/VestingWallet.sol#L112-L116

- ID-20

Reentrancy in [BasePayment.withdraw(address,uint256,uint8,bytes32,bytes32,address)](contracts/solidity/payments/BasePayment.sol#L24-L41):
External calls:
- [withdrawalToken.approve(router,amount)](contracts/solidity/payments/BasePayment.sol#L34)
- [IProcessingPayments(paymentsEngine).payERC(stealth,path,type()(uint256).max,amount,type()(uint256).max)](contracts/solidity/payments/BasePayment.sol#L39)
State variables written after the call(s):
- [stealth = newStealth](contracts/solidity/payments/BasePayment.sol#L40)
[BasePayment.stealth](contracts/solidity/payments/BasePayment.sol#L9) can be used in cross function reentrancies:
- [BasePayment.checkWithdrawalRequest(address,uint8,bytes32,bytes32,address)](contracts/solidity/payments/BasePayment.sol#L44-L51)
- [BasePayment.initStealth(address)](contracts/solidity/payments/BasePayment.sol#L13-L17)
- [BasePayment.initToken(address)](contracts/solidity/payments/BasePayment.sol#L19-L22)
- [BasePayment.stealth](contracts/solidity/payments/BasePayment.sol#L9)
- [BasePayment.withdraw(address,uint256,uint8,bytes32,bytes32,address)](contracts/solidity/payments/BasePayment.sol#L24-L41)

contracts/solidity/payments/BasePayment.sol#L24-L41

- ID-21

Reentrancy in [PullPayment.unpause(uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L27-L33):
 External calls:
 - [Notifier(notifier).notifyUnpaused()](contracts/solidity/payments/PullPayment.sol#L31)
 State variables written after the call(s):
 - [stealth = newStealth](contracts/solidity/payments/PullPayment.sol#L32)
 [BasePayment.stealth](contracts/solidity/payments/BasePayment.sol#L9) can be used in cross function reentrancies:
 - [BasePayment.checkWithdrawalRequest(address,uint8,bytes32,bytes32,address)]
(contracts/solidity/payments/BasePayment.sol#L44-L51)
 - [PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61)
 - [BasePayment.initStealth(address)](contracts/solidity/payments/BasePayment.sol#L13-L17)
 - [BasePayment.initToken(address)](contracts/solidity/payments/BasePayment.sol#L19-L22)
 - [PullPayment.pause(uint8,bytes32,bytes32)](contracts/solidity/payments/PullPayment.sol#L20-L25)
 - [BasePayment.stealth](contracts/solidity/payments/BasePayment.sol#L9)
 - [PullPayment.unpause(uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L27-L33)
 - [BasePayment.withdraw(address,uint256,uint8,bytes32,bytes32,address)]
(contracts/solidity/payments/BasePayment.sol#L24-L41)

contracts/solidity/payments/PullPayment.sol#L27-L33

- ID-22

Reentrancy in [VestingWallet.updateRewardsFarm(address)](contracts/solidity/vesting/VestingWallet.sol#L106-L110):
 External calls:
 - [IERC20(smarty).approve(newFarm,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L107)
 - [_beforeUpdateFarm(rewardsFarm,newFarm)](contracts/solidity/vesting/VestingWallet.sol#L108)
 - [(returnData) = oldFarm.call(payload)](contracts/solidity/vesting/VestingWallet.sol#L121)
 - [(None,returnData) = oldFarm.call(payload)](contracts/solidity/vesting/VestingWallet.sol#L125)
 - [IStakingFarm(newFarm).stake(stakedAmount)](contracts/solidity/vesting/VestingWallet.sol#L129)
 State variables written after the call(s):
 - [rewardsFarm = newFarm](contracts/solidity/vesting/VestingWallet.sol#L109)
 [VestingWallet.rewardsFarm](contracts/solidity/vesting/VestingWallet.sol#L19) can be used in cross function reentrancies:
 - [VestingWallet.burnRewardsFarmToken(uint256)](contracts/solidity/vesting/VestingWallet.sol#L69-L71)
 - [VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56)
 - [VestingWallet.rewardsFarm](contracts/solidity/vesting/VestingWallet.sol#L19)
 - [VestingWallet.stakeToRewardsFarm(uint256)](contracts/solidity/vesting/VestingWallet.sol#L64-L67)
 - [VestingWallet.updateRewardsFarm(address)](contracts/solidity/vesting/VestingWallet.sol#L106-L110)
 - [VestingWallet.withdrawFromRewardsFarm(uint256)](contracts/solidity/vesting/VestingWallet.sol#L73-L75)

contracts/solidity/vesting/VestingWallet.sol#L106-L110

uninitialized-local

Uninitialized local variables

Configuration

- Check: `uninitialized-local`
- Severity: `Medium`
- Confidence: `Medium`

Description

Uninitialized local variables.

Exploit Scenario:

```
contract Uninitialized is Owner{
    function withdraw() payable public onlyOwner{
        address to;
        to.transfer(this.balance)
    }
}
```

Bob calls `transfer`. As a result, all Ether is sent to the address `0x0` and is lost.

Recommendation

Initialize all the variables. If a variable is meant to be initialized to zero, explicitly set it to zero to improve code readability.

- ID-23
[LiquidityFarm.stakeToken(address,uint256,bool).amount0Out](contracts/solidity/rewards/LiquidityFarm.sol#L35) is a local variable never initialized

contracts/solidity/rewards/LiquidityFarm.sol#L35

- ID-24
[LiquidityFarm.stakeToken(address,uint256,bool).amount1Out](contracts/solidity/rewards/LiquidityFarm.sol#L36) is a local variable never initialized

contracts/solidity/rewards/LiquidityFarm.sol#L36

- ID-25
[UniswapV2Library.getAmountsOut(address,uint256,address[]).i](contracts/solidity/uniswap/UniswapV2Library.sol#L80) is a local variable never initialized

contracts/solidity/uniswap/UniswapV2Library.sol#L80

- ID-26
[UniswapV2Router._swap(uint256[],address[],address).i](contracts/solidity/uniswap/UniswapV2Router.sol#L50) is a local variable never initialized

contracts/solidity/uniswap/UniswapV2Router.sol#L50

- ID-27
[MerchantWallet.distribute(address,address[],uint256[]).i](contracts/solidity/merchant/MerchantWallet.sol#L58) is a local variable never initialized

contracts/solidity/merchant/MerchantWallet.sol#L58

- ID-28
[MerchantWallet.distribute(address,address[],uint256[]).i_scope_0](contracts/solidity/merchant/MerchantWallet.sol#L67) is a local variable never initialized

contracts/solidity/merchant/MerchantWallet.sol#L67

- ID-29
[CustomerWallet.migrate(address,uint8,bytes32,bytes32).index](contracts/solidity/CustomerWallet.sol#L97) is a local variable never initialized

contracts/solidity/CustomerWallet.sol#L97

- ID-30
[BadToken.transferFrom(address,address,uint256).result](contracts/solidity/test/BadToken.sol#L22) is a local variable never initialized

contracts/solidity/test/BadToken.sol#L22

- ID-31
[SmartyProcessingWithdrawals.withdraw(Structs.WithdrawalInput,Structs.WithdrawalOutput).k](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L111) is a local variable never initialized

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L111

- ID-32
[SmartyProcessingPayments.internalPay(address,address[],uint256,int256,address,uint256,uint256,IFinanceRouter,bool).i](contracts/solidity/payments/SmartyProcessingPayments.sol#L235) is a local variable never initialized

contracts/solidity/payments/SmartyProcessingPayments.sol#L235

- ID-33
[SmartyProcessingPayments.checkSimplePayment(address[],address[][],uint256[],int256[],uint8[],bytes32[],bytes32[],uint256[],uint256).nextNum](contracts/solidity/payments/SmartyProcessingPayments.sol#L156) is a local variable never initialized

contracts/solidity/payments/SmartyProcessingPayments.sol#L156

- ID-34
[BadToken.transfer(address,uint256).result](contracts/solidity/test/BadToken.sol#L16) is a local variable never initialized

contracts/solidity/test/BadToken.sol#L16

- ID-35
[BatchOfBalances.erc20Balances(address,address[]).i](contracts/solidity/infrastructure/BatchOfBalances.sol#L12) is a local variable never initialized

contracts/solidity/infrastructure/BatchOfBalances.sol#L12

- ID-36
[BatchOfBalances.ethBalances(address[]).i](contracts/solidity/infrastructure/BatchOfBalances.sol#L20) is a local variable never initialized

contracts/solidity/infrastructure/BatchOfBalances.sol#L20

- ID-37
[VestingWallet._beforeUpdateFarm(address,address).stakedAmount](contracts/solidity/vesting/VestingWallet.sol#L119) is a local variable never initialized

contracts/solidity/vesting/VestingWallet.sol#L119

unused-return

Unused return

Configuration

- Check: `unused-return`
- Severity: `Medium`
- Confidence: `Medium`

Description

The return value of an external call is not stored in a local or state variable.

Exploit Scenario:

```
contract MyConc{
    using SafeMath for uint;
    function my_func(uint a, uint b) public{
        a.add(b);
    }
}
```

`MyConc` calls `add` of `SafeMath`, but does not store the result in `a`. As a result, the computation has no effect.

Recommendation

Ensure that all the return values of the function calls are used.

- ID-38
[VestingWallet._beforeUpdateFarm(address,address)](contracts/solidity/vesting/VestingWallet.sol#L118-L131) ignores return value by [IStakingFarm(newFarm).stake(stakedAmount)](contracts/solidity/vesting/VestingWallet.sol#L129)

contracts/solidity/vesting/VestingWallet.sol#L118-L131

- ID-39
[VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56) ignores return value by [IERC20(ILiquidityFarm(liquidityFarm).stakedToken()).approve(liquidityFarm,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L54)

contracts/solidity/vesting/VestingWallet.sol#L33-L56

- ID-40
[BasePayment.withdraw(address,uint256,uint8,bytes32,bytes32,address)](contracts/solidity/payments/BasePayment.sol#L24-L41) ignores return value by [withdrawalToken.approve(router,amount)](contracts/solidity/payments/BasePayment.sol#L34)

contracts/solidity/payments/BasePayment.sol#L24-L41

- ID-41
[VestingWallet.updateRewardsFarm(address)](contracts/solidity/vesting/VestingWallet.sol#L106-L110) ignores return value by [IERC20(smarty).approve(newFarm,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L107)

contracts/solidity/vesting/VestingWallet.sol#L106-L110

- ID-42
[VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56) ignores return value by [IERC20(smarty).approve(uniswapV2Router02,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L52)

contracts/solidity/vesting/VestingWallet.sol#L33-L56

- ID-43

[VestingWallet.withdrawFromLiquidityFarm(uint256)](contracts/solidity/vesting/VestingWallet.sol#L97-L101) ignores return value by [IUniswapV2Router02(uniswapV2Router02).removeLiquidityETH(smarty,stakedTokenAmount,0x1,0x1,address(this),block.timestamp)](contracts/solidity/vesting/VestingWallet.sol#L99)

contracts/solidity/vesting/VestingWallet.sol#L97-L101

- ID-44

[PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61) ignores return value by [withdrawalToken.approve(router,balance)](contracts/solidity/payments/PullPayment.sol#L51)

contracts/solidity/payments/PullPayment.sol#L40-L61

- ID-45

[SmartyCryptoProcessingDelegate.isBlackListed(address,address)](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L87-L89) ignores return value by [processingManager.isBlackListed(token,stealth)](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L88)

contracts/solidity/SmartyCryptoProcessingDelegate.sol#L87-L89

- ID-46

[VestingWallet.stakeToRewardsFarm(uint256)](contracts/solidity/vesting/VestingWallet.sol#L64-L67) ignores return value by [IRewardsFarm(rewardsFarm).stake(smartyAmount)](contracts/solidity/vesting/VestingWallet.sol#L66)

contracts/solidity/vesting/VestingWallet.sol#L64-L67

- ID-47

[VestingWallet.burnRewardsFarmToken(uint256)](contracts/solidity/vesting/VestingWallet.sol#L69-L71) ignores return value by [IRewardsFarm(rewardsFarm).burn(farmTokenAmount,address(this))](contracts/solidity/vesting/VestingWallet.sol#L70)

contracts/solidity/vesting/VestingWallet.sol#L69-L71

- ID-48

[MultichainRewardsController.updateRewardsToken(address,address)](contracts/solidity/multichain/MultichainRewardsController.sol#L92-L98) ignores return value by [IERC20(rewardsToken).approve(multichainRouter,type()(uint256).max)](contracts/solidity/multichain/MultichainRewardsController.sol#L97)

contracts/solidity/multichain/MultichainRewardsController.sol#L92-L98

- ID-49

[VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56) ignores return value by [IERC20(smarty).approve(rewardsFarm,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L51)

contracts/solidity/vesting/VestingWallet.sol#L33-L56

- ID-50

[VestingWallet._burnLiquidityFarmToken(uint256)](contracts/solidity/vesting/VestingWallet.sol#L88-L95) ignores return value by [IUniswapV2Router02(uniswapV2Router02).removeLiquidityETH(smarty,stakedTokenAmount,0x1,0x1,address(this),block.timestamp)](contracts/solidity/vesting/VestingWallet.sol#L93)

contracts/solidity/vesting/VestingWallet.sol#L88-L95

- ID-51

[BSCRewardsController.distributeEth()](contracts/solidity/bsc/BSCRewardsController.sol#L36-L46) ignores return value by [UniswapV2Router(address(this)).swapExactETHForTokens{value: eth}(0x1,path,rewardsFarm,type()(uint256).max)](contracts/solidity/bsc/BSCRewardsController.sol#L43)

contracts/solidity/bsc/BSCRewardsController.sol#L36-L46

- ID-52

[SmartyTransfer.withdrawFunds(address,address,uint256,bytes)](contracts/solidity/libraries/SmartyTransfer.sol#L23-L42) ignores return value by [IERC20(token).approve(multichainRouter,amount)](contracts/solidity/libraries/SmartyTransfer.sol#L38)

contracts/solidity/libraries/SmartyTransfer.sol#L23-L42

- ID-53

[VestingWallet.stakeToLiquidityFarm(uint256)](contracts/solidity/vesting/VestingWallet.sol#L77-L82) ignores return value by [ILiquidityFarm(liquidityFarm).stake(liquidity)](contracts/solidity/vesting/VestingWallet.sol#L80)

contracts/solidity/vesting/VestingWallet.sol#L77-L82

- ID-54

[VestingWallet.updateLiquidityFarm(address)](contracts/solidity/vesting/VestingWallet.sol#L112-L116) ignores return value by [IERC20(ILiquidityFarm(newFarm).stakedToken()).approve(newFarm,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L113)

contracts/solidity/vesting/VestingWallet.sol#L112-L116

- ID-55

[VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56) ignores return value by [IERC20(ILiquidityFarm(liquidityFarm).stakedToken()).approve(uniswapV2Router02,type()(uint256).max)](contracts/solidity/vesting/VestingWallet.sol#L55)

contracts/solidity/vesting/VestingWallet.sol#L33-L56

shadowing-local

Local variable shadowing

Configuration

- Check: `shadowing-local`
- Severity: `Low`
- Confidence: `High`

Description

Detection of shadowing using local variables.

Exploit Scenario:

```
pragma solidity ^0.4.24;

contract Bug {
    uint owner;

    function sensitive_function(address owner) public {
        // ...
        require(owner == msg.sender);
    }

    function alternate_sensitive_function() public {
        address owner = msg.sender;
        // ...
        require(owner == msg.sender);
    }
}
```

`sensitive_function.owner` shadows `Bug.owner`. As a result, the use of `owner` in `sensitive_function` might be incorrect.

Recommendation

Rename the local variables that shadow another component.

- ID-56
[SmartyCryptoProcessingDelegate.router().router](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L120) shadows:
 - [SmartyCryptoProcessingDelegate.router()](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L120-L122) (function)
 - [PaymentsAPI.router()](contracts/solidity/payments/interfaces/PaymentsAPI.sol#L9) (function)

contracts/solidity/SmartyCryptoProcessingDelegate.sol#L120

- ID-57
[BaseFarm.constructor(address,string,string).symbol](contracts/solidity/rewards/BaseFarm.sol#L11) shadows:
 - [ERC20.symbol()](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L70-L72) (function)
 - [IERC20Metadata.symbol()](node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#L22) (function)

contracts/solidity/rewards/BaseFarm.sol#L11

- ID-58
[PaymentsAPI.router().router](contracts/solidity/payments/interfaces/PaymentsAPI.sol#L9) shadows:
 - [PaymentsAPI.router()](contracts/solidity/payments/interfaces/PaymentsAPI.sol#L9) (function)

contracts/solidity/payments/interfaces/PaymentsAPI.sol#L9

- ID-59

[BaseFarm.stakedTokenAmountAfterBurning(uint256).totalSupply](contracts/solidity/rewards/BaseFarm.sol#L62) shadows:
 - [ERC20.totalSupply()](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L94-L96) (function)
 - [IERC20.totalSupply()](node_modules/@openzeppelin/contracts/token/ERC20/IERC20.sol#L13) (function)

contracts/solidity/rewards/BaseFarm.sol#L62

- ID-60

[SmartyCryptoProcessing._delegate(address).implementation](contracts/solidity/SmartyCryptoProcessing.sol#L60) shadows:
 - [SmartyCryptoProcessing.implementation()](contracts/solidity/SmartyCryptoProcessing.sol#L48-L50) (function)

contracts/solidity/SmartyCryptoProcessing.sol#L60

- ID-61

[BaseFarm.constructor(address,string,string).name](contracts/solidity/rewards/BaseFarm.sol#L11) shadows:
 - [ERC20.name()](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L62-L64) (function)
 - [IERC20Metadata.name()](node_modules/@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol#L17) (function)

contracts/solidity/rewards/BaseFarm.sol#L11

events-access

Missing events access control

Configuration

- Check: `events-access`
- Severity: `Low`
- Confidence: `Medium`

Description

Detect missing events for critical access control parameters

Exploit Scenario:

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

`updateOwner()` has no event, so it is difficult to track off-chain owner changes.

Recommendation

Emit an event for critical parameter changes.

- ID-62
[VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56) should emit an event for:
 - [investor = _investor](contracts/solidity/vesting/VestingWallet.sol#L43)
 - [owner = tx.origin](contracts/solidity/vesting/VestingWallet.sol#L49)

contracts/solidity/vesting/VestingWallet.sol#L33-L56

events-maths

Missing events arithmetic

Configuration

- Check: `events-maths`
- Severity: `Low`
- Confidence: `Medium`

Description

Detect missing events for critical arithmetic parameters.

Exploit Scenario:

```
contract C {

    modifier onlyOwner {
        if (msg.sender != owner) throw;
        _;
    }

    function setBuyPrice(uint256 newBuyPrice) onlyOwner public {
        buyPrice = newBuyPrice;
    }

    function buy() external {
     ... // buyPrice is used to determine the number of tokens purchased
    }
}
```

`setBuyPrice()` does not emit an event, so it is difficult to track changes in the value of `buyPrice` off-chain.

Recommendation

Emit an event for critical parameter changes.

- ID-63
[MultichainRewardsController.updateDestChainId(uint256)]
(contracts/solidity/multichain/MultichainRewardsController.sol#L81-L84) should emit an event for:
 - [destChainId = newDestChainId](contracts/solidity/multichain/MultichainRewardsController.sol#L83)

contracts/solidity/multichain/MultichainRewardsController.sol#L81-L84

- ID-64
[BaseRewardsController.setMinimalDistribution(uint256)](contracts/solidity/rewards/BaseRewardsController.sol#L54-L56)
should emit an event for:
 - [minimumDistribute = newValue](contracts/solidity/rewards/BaseRewardsController.sol#L55)

contracts/solidity/rewards/BaseRewardsController.sol#L54-L56

missing-zero-check

Missing zero address validation

Configuration

- Check: `missing-zero-check`
- Severity: `Low`
- Confidence: `Medium`

Description

Detect missing zero address validation.

Exploit Scenario:

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

Bob calls `updateOwner` without specifying the `newOwner`, so Bob loses ownership of the contract.

Recommendation

Check that the address is not zero.

- ID-65
[BSCCustomerWallet.getEth(uint256,uint256,uint256,address,uint8,bytes32,bytes32,address[],uint256).sender]
(contracts/solidity/bsc/BSCCustomerWallet.sol#L21) lacks a zero-check on :
 - [sender.transfer(senderFee)](contracts/solidity/bsc/BSCCustomerWallet.sol#L37)

contracts/solidity/bsc/BSCCustomerWallet.sol#L21

- ID-66
[SmartyProcessingWithdrawals.constructor(address,address)._invoicesStorage]
(contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L27) lacks a zero-check on :
 - [invoicesStorage = _invoicesStorage](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L28)

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L27

- ID-67
[SafeTransfer.constructor(address).wallet](contracts/solidity/onramp/SafeTransfer.sol#L16) lacks a zero-check on :
 - [transferWallet = wallet](contracts/solidity/onramp/SafeTransfer.sol#L17)

contracts/solidity/onramp/SafeTransfer.sol#L16

- ID-68
[SmartyCryptoProcessingDelegate.constructor(address,address,address)._withdrawalsEngine]
(contracts/solidity/SmartyCryptoProcessingDelegate.sol#L22) lacks a zero-check on :
 - [withdrawalsEngine = _withdrawalsEngine](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L24)

contracts/solidity/SmartyCryptoProcessingDelegate.sol#L22

- ID-69

[SmartyProcessingPayments.constructor(address,address,address,address,address,address)._merchantWalletFactory] (contracts/solidity/payments/SmartyProcessingPayments.sol#L24) lacks a zero-check on :
 - [merchantWalletFactory = _merchantWalletFactory](contracts/solidity/payments/SmartyProcessingPayments.sol#L27)

contracts/solidity/payments/SmartyProcessingPayments.sol#L24

- ID-70

[BSCRewardsController.updateRewardsToken(address).newRewardsToken] (contracts/solidity/bsc/BSCRewardsController.sol#L15) lacks a zero-check on :
 - [rewardsToken = newRewardsToken](contracts/solidity/bsc/BSCRewardsController.sol#L16)

contracts/solidity/bsc/BSCRewardsController.sol#L15

- ID-71

[VestingWallet.constructor(address)._vestingWalletFactory](contracts/solidity/vesting/VestingWallet.sol#L24) lacks a zero-check on :
 - [vestingWalletFactory = _vestingWalletFactory](contracts/solidity/vesting/VestingWallet.sol#L25)

contracts/solidity/vesting/VestingWallet.sol#L24

- ID-72

[BasePayment.initToken(address)._token](contracts/solidity/payments/BasePayment.sol#L19) lacks a zero-check on :
 - [token = _token](contracts/solidity/payments/BasePayment.sol#L21)

contracts/solidity/payments/BasePayment.sol#L19

- ID-73

[SafeTransfer.forwardNative(bytes32,address).recipient](contracts/solidity/onramp/SafeTransfer.sol#L27) lacks a zero-check on :
 - [recipient.transfer(msg.value)](contracts/solidity/onramp/SafeTransfer.sol#L30)

contracts/solidity/onramp/SafeTransfer.sol#L27

- ID-74

[BadToken.constructor(address)._allowed](contracts/solidity/test/BadToken.sol#L10) lacks a zero-check on :
 - [allowed = _allowed](contracts/solidity/test/BadToken.sol#L11)

contracts/solidity/test/BadToken.sol#L10

- ID-75

[UniswapV2Router.constructor(address,address,address)._WETH](contracts/solidity/uniswap/UniswapV2Router.sol#L14) lacks a zero-check on :
 - [WETH = _WETH](contracts/solidity/uniswap/UniswapV2Router.sol#L17)

contracts/solidity/uniswap/UniswapV2Router.sol#L14

- ID-76

[BaseSimplePayment.initToken(address)._token](contracts/solidity/payments/simple/BaseSimplePayment.sol#L26) lacks a zero-check on :
 - [token = _token](contracts/solidity/payments/simple/BaseSimplePayment.sol#L28)

contracts/solidity/payments/simple/BaseSimplePayment.sol#L26

- ID-77

[MerchantWallet.constructor(address,address)._processing](contracts/solidity/merchant/MerchantWallet.sol#L19) lacks a zero-check on :
 - [processing = _processing](contracts/solidity/merchant/MerchantWallet.sol#L20)

contracts/solidity/merchant/MerchantWallet.sol#L19

- ID-78

[BasePayment.withdraw(address,uint256,uint8,bytes32,bytes32,address).newStealth] (contracts/solidity/payments/BasePayment.sol#L24) lacks a zero-check on :
 - [stealth = newStealth](contracts/solidity/payments/BasePayment.sol#L40)

contracts/solidity/payments/BasePayment.sol#L24

- ID-79

[SmartyCryptoProcessingDelegate.constructor(address,address,address)._paymentsEngine] (contracts/solidity/SmartyCryptoProcessingDelegate.sol#L22) lacks a zero-check on :
 - [paymentsEngine = _paymentsEngine](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L23)

contracts/solidity/SmartyCryptoProcessingDelegate.sol#L22

- ID-80

[PushPayment.withdrawEth(address,uint256,uint8,bytes32,bytes32,address).newStealth] (contracts/solidity/payments/PushPayment.sol#L19) lacks a zero-check on :
 - [stealth = newStealth](contracts/solidity/payments/PushPayment.sol#L28)

contracts/solidity/payments/PushPayment.sol#L19

- ID-81

[VestingWallet.updateLiquidityFarm(address).newFarm](contracts/solidity/vesting/VestingWallet.sol#L112) lacks a zero-check on :
 - [liquidityFarm = newFarm](contracts/solidity/vesting/VestingWallet.sol#L115)

contracts/solidity/vesting/VestingWallet.sol#L112

- ID-82

[SmartyProcessingPayments.constructor(address,address,address,address,address,address)._invoicesStorage] (contracts/solidity/payments/SmartyProcessingPayments.sol#L24) lacks a zero-check on :
 - [invoicesStorage = _invoicesStorage](contracts/solidity/payments/SmartyProcessingPayments.sol#L25)

contracts/solidity/payments/SmartyProcessingPayments.sol#L24

- ID-83

[VestingWallet.updateRewardsFarm(address).newFarm](contracts/solidity/vesting/VestingWallet.sol#L106) lacks a zero-check on :
 - [rewardsFarm = newFarm](contracts/solidity/vesting/VestingWallet.sol#L109)

contracts/solidity/vesting/VestingWallet.sol#L106

- ID-84

[MerchantWallet.constructor(address,address)._walletFactory](contracts/solidity/merchant/MerchantWallet.sol#L19) lacks a zero-check on :
 - [walletFactory = _walletFactory](contracts/solidity/merchant/MerchantWallet.sol#L21)

contracts/solidity/merchant/MerchantWallet.sol#L19

- ID-85

[SmartyProcessingPayments.constructor(address,address,address,address,address,address)._walletFactory] (contracts/solidity/payments/SmartyProcessingPayments.sol#L24) lacks a zero-check on :
 - [walletFactory = _walletFactory](contracts/solidity/payments/SmartyProcessingPayments.sol#L26)

contracts/solidity/payments/SmartyProcessingPayments.sol#L24

- ID-86

[SmartyProcessingPayments.constructor(address,address,address,address,address,address)._customerWallet] (contracts/solidity/payments/SmartyProcessingPayments.sol#L24) lacks a zero-check on :
 - [customerWallet = _customerWallet](contracts/solidity/payments/SmartyProcessingPayments.sol#L28)

contracts/solidity/payments/SmartyProcessingPayments.sol#L24

calls-loop

Calls inside a loop

Configuration

- Check: `calls-loop`
- Severity: `Low`
- Confidence: `Medium`

Description

Calls inside a loop might lead to a denial-of-service attack.

Exploit Scenario:

```
contract CallsInLoop{

    address[] destinations;

    constructor(address[] newDestinations) public{
        destinations = newDestinations;
    }

    function bad() external{
        for (uint i=0; i < destinations.length; i++){
            destinations[i].transfer(i);
        }
    }

}
```

If one of the destinations has a fallback function that reverts, `bad` will always revert.

Recommendation

Favor pull over push strategy for external calls.

- ID-87
[SmartyTransfer.withdrawFunds(address,address,uint256,bytes)](contracts/solidity/libraries/SmartyTransfer.sol#L23-L42) has external calls inside a loop: [require(bool,string)(IERC20(token).transfer(destination,amount),NCPS-04.011:TRANSFER_FAILED)](contracts/solidity/libraries/SmartyTransfer.sol#L30)

contracts/solidity/libraries/SmartyTransfer.sol#L23-L42

- ID-88
[SmartyProcessingWithdrawals.withdraw(Structs.WithdrawalInput,Structs.WithdrawalOutput)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139) has external calls inside a loop: [paymentAmount = invoicesManager.getInvoiceAmount(key)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L100)

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139

- ID-89
[BatchOfBalances.erc20Balances(address,address[])](contracts/solidity/infrastructure/BatchOfBalances.sol#L8-L15) has external calls inside a loop: [balances[i] = token.balanceOf(subjects[i])](contracts/solidity/infrastructure/BatchOfBalances.sol#L13)

contracts/solidity/infrastructure/BatchOfBalances.sol#L8-L15

- ID-90
[CustomerWallet.migrate(address,uint8,bytes32,bytes32)](contracts/solidity/CustomerWallet.sol#L85-L105) has external calls inside a loop: [balance = IERC20(tokens[index]).balanceOf(address(this))](contracts/solidity/CustomerWallet.sol#L99)

contracts/solidity/CustomerWallet.sol#L85-L105

- ID-91

[PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61) has external calls inside a loop: [balance = withdrawalToken.balanceOf(address(this))] (contracts/solidity/payments/PullPayment.sol#L49)

contracts/solidity/payments/PullPayment.sol#L40-L61

- ID-92

[SmartyTransfer.withdrawFunds(address,address,uint256,bytes)](contracts/solidity/libraries/SmartyTransfer.sol#L23-L42) has external calls inside a loop: [MultichainRouter(multichainRouter).anySwapOutUnderlying(wrappedToken,destination,amount,chainId)] (contracts/solidity/libraries/SmartyTransfer.sol#L39)

contracts/solidity/libraries/SmartyTransfer.sol#L23-L42

- ID-93

[SmartyProcessingPayments.internalPay(address,address[],uint256,int256,address,uint256,uint256,IFinanceRouter,bool)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L230-L249) has external calls inside a loop: [IInvoicesStorageManager(invoicesStorage).storeInvoice(key,amountOut)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L243)

contracts/solidity/payments/SmartyProcessingPayments.sol#L230-L249

- ID-94

[KYC.check(address,address)](contracts/solidity/ERC20/KYC.sol#L15-L17) has external calls inside a loop: [clean = ! processingManager.isBlackListed(token,customer)](contracts/solidity/ERC20/KYC.sol#L16)

contracts/solidity/ERC20/KYC.sol#L15-L17

- ID-95

[SmartyTransfer.withdrawFunds(address,address,uint256,bytes)](contracts/solidity/libraries/SmartyTransfer.sol#L23-L42) has external calls inside a loop: [require(bool,string)(MultichainToken(wrappedToken).underlying() == token,NCPS-04.013:BAD_WRAPPED_TOKEN)](contracts/solidity/libraries/SmartyTransfer.sol#L37)

contracts/solidity/libraries/SmartyTransfer.sol#L23-L42

- ID-96

[SmartyProcessingPayments.multiDelegatePay(Structs.DelegatePayment[])] (contracts/solidity/payments/SmartyProcessingPayments.sol#L213-L228) has external calls inside a loop: [require(bool,string) (processingManager.tokenSupported(currentToken),NCPS-03.021:TOKEN_NOT_SUPPORTED)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L219)

contracts/solidity/payments/SmartyProcessingPayments.sol#L213-L228

- ID-97

[SmartyProcessingWithdrawals.withdraw(Structs.WithdrawalInput,Structs.WithdrawalOutput)] (contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139) has external calls inside a loop: [withdraw(output.tokenOutputs[index].token,processingManager.rewardsContract(),output.tokenOutputs[index].fee,abi.encode(block.chainid))](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L134-L135)

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139

- ID-98

[SmartyProcessingPayments.isPaid(address)](contracts/solidity/payments/SmartyProcessingPayments.sol#L272-L274) has external calls inside a loop: [IInvoicesStorageManager(invoicesStorage).getInvoiceAmount(key) != 0] (contracts/solidity/payments/SmartyProcessingPayments.sol#L273)

contracts/solidity/payments/SmartyProcessingPayments.sol#L272-L274

- ID-99

[BaseSimplePayment.flush(address)](contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38) has external calls inside a loop: [require(bool,string)(address(merchantWallet).send(address(this).balance),NCPS-38.021:TRANSFER_FAILED)] (contracts/solidity/payments/simple/BaseSimplePayment.sol#L36)

contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38

- ID-100

[PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61) has external calls inside a loop: [withdrawalToken.approve(router,balance)](contracts/solidity/payments/PullPayment.sol#L51)

contracts/solidity/payments/PullPayment.sol#L40-L61

- ID-101

[PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61) has external calls inside a loop: [router = address(PaymentsAPI(paymentsEngine).router())] (contracts/solidity/payments/PullPayment.sol#L47)

contracts/solidity/payments/PullPayment.sol#L40-L61

- ID-102

[KYC.check(address,address)](contracts/solidity/ERC20/KYC.sol#L15-L17) has external calls inside a loop: [processingManager.kycEnabled()](contracts/solidity/ERC20/KYC.sol#L16)

contracts/solidity/ERC20/KYC.sol#L15-L17

- ID-103

[SmartyTransfer.withdrawFunds(address,address,uint256,bytes)](contracts/solidity/libraries/SmartyTransfer.sol#L23-L42) has external calls inside a loop: [IERC20(token).approve(multichainRouter,amount)] (contracts/solidity/libraries/SmartyTransfer.sol#L38)

contracts/solidity/libraries/SmartyTransfer.sol#L23-L42

- ID-104

[SmartyTransfer.transferFunds(address[],address,address,uint256,uint256,uint256,IFinanceRouter)] (contracts/solidity/libraries/SmartyTransfer.sol#L11-L20) has external calls inside a loop: [financeRouter.swapTokensForExactTokens(amountInMax,amountOut,path,sender,destination,deadline)] (contracts/solidity/libraries/SmartyTransfer.sol#L17)

contracts/solidity/libraries/SmartyTransfer.sol#L11-L20

- ID-105

[SmartyProcessingWithdrawals.withdraw(Structs.WithdrawalInput,Structs.WithdrawalOutput)] (contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139) has external calls inside a loop: [invoicesManager.storeInvoice(key,- 1)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L103)

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139

- ID-106

[BaseSimplePayment.flush(address)](contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38) has external calls inside a loop: [merchantWallet = BaseMinimalProxyFactory(merchantWalletFactory).calcSaltedAddress(keccak256(bytes) (abi.encodePacked(merchant)),ProcessingManagerAPI(processing).merchantWallet())] (contracts/solidity/payments/simple/BaseSimplePayment.sol#L32)

contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38

- ID-107

[BaseSimplePayment.flush(address)](contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38) has external calls inside a loop: [require(bool,string)(IERC20(flushedToken).transfer(merchantWallet,IERC20(flushedToken).balanceOf(address(this))),NCPS-38.021:TRANSFER_FAILED)](contracts/solidity/payments/simple/BaseSimplePayment.sol#L34)

contracts/solidity/payments/simple/BaseSimplePayment.sol#L31-L38

- ID-108

[SmartyProcessingPayments.delegatePay(Structs.DelegatePayment,bool,bytes32)](contracts/solidity/payments/SmartyProcessingPayments.sol#L196-L210) has external calls inside a loop: [wallet = address(BaseMinimalProxyFactory(walletFactory).getInstance(keccak256(bytes)(abi.encodePacked(recovered)),customerWallet))](contracts/solidity/payments/SmartyProcessingPayments.sol#L200)

contracts/solidity/payments/SmartyProcessingPayments.sol#L196-L210

- ID-109

[SmartyTransfer.withdrawFunds(address,address,uint256,bytes)](contracts/solidity/libraries/SmartyTransfer.sol#L23-L42) has external calls inside a loop: [address(destination).transfer(amount)](contracts/solidity/libraries/SmartyTransfer.sol#L28)

contracts/solidity/libraries/SmartyTransfer.sol#L23-L42

- ID-110

[SmartyTransfer.transferFunds(address[],address,address,uint256,uint256,uint256,IFinanceRouter)](contracts/solidity/libraries/SmartyTransfer.sol#L11-L20) has external calls inside a loop: [financeRouter.pay(targetToken,sender,destination,amountOut)](contracts/solidity/libraries/SmartyTransfer.sol#L15)

contracts/solidity/libraries/SmartyTransfer.sol#L11-L20

- ID-111

[SmartyProcessingWithdrawals.withdraw(Structs.WithdrawalInput,Structs.WithdrawalOutput)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139) has external calls inside a loop: [processingManager.triggerRewardsController(output.tokenOutputs[index].token)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L136)

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139

- ID-112

[CustomerWallet.migrate(address,uint8,bytes32,bytes32)](contracts/solidity/CustomerWallet.sol#L85-L105) has external calls inside a loop: [require(bool,string)(IERC20(tokens[index]).transfer(newWallet,balance),NCPS-01.074:TRANSFER_FAILED)](contracts/solidity/CustomerWallet.sol#L101)

contracts/solidity/CustomerWallet.sol#L85-L105

- ID-113

[PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61) has external calls inside a loop: [IProcessingPayments(paymentsEngine).payERC(stealth,path,type()(uint256).max,balance,type()(uint256).max)](contracts/solidity/payments/PullPayment.sol#L56)

contracts/solidity/payments/PullPayment.sol#L40-L61

- ID-114

[PullPayment.destroy(address[],uint8,bytes32,bytes32,address)](contracts/solidity/payments/PullPayment.sol#L40-L61) has external calls inside a loop: [balance > withdrawalToken.allowance(address(this),router)](contracts/solidity/payments/PullPayment.sol#L50)

contracts/solidity/payments/PullPayment.sol#L40-L61

reentrancy-benign

Reentrancy vulnerabilities

Configuration

- Check: `reentrancy-benign`
- Severity: `Low`
- Confidence: `Medium`

Description

Detection of the reentrancy bug. Only report reentrancy that acts as a double call (see `reentrancy-eth`, `reentrancy-no-eth`).

Exploit Scenario:

```
function callme(){
    if( ! (msg.sender.call()() ) ){
        throw;
    }
    counter += 1
}
```

`callme` contains a reentrancy. The reentrancy is benign because it's exploitation would have the same effect as two consecutive calls.

Recommendation

Apply the `check-effects-interactions` pattern.

- ID-115

Reentrancy in [BaseFarm.stake(uint256)](contracts/solidity/rewards/BaseFarm.sol#L16-L30):
External calls:
- [require(bool,string)(IERC20(stakedToken).transferFrom(msg.sender,address(this),amount),NCPS-05.013:TRANSFER_FAILED)](contracts/solidity/rewards/BaseFarm.sol#L23)
State variables written after the call(s):
- [_mint(msg.sender,farmTokenAmount)](contracts/solidity/rewards/BaseFarm.sol#L27)
 - [_balances[account] += amount](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L263)
- [_mint(msg.sender,farmTokenAmount)](contracts/solidity/rewards/BaseFarm.sol#L27)
 - [_totalSupply += amount](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L262)

contracts/solidity/rewards/BaseFarm.sol#L16-L30

- ID-116

Reentrancy in [LiquidityFarm.stakeToken(address,uint256,bool)](contracts/solidity/rewards/LiquidityFarm.sol#L24-L73):
External calls:
- [IERC20(token).transferFrom(msg.sender,stakedToken,swapAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L49)
- [pair.swap(amount0Out,amount1Out,address(this),new bytes(0))](contracts/solidity/rewards/LiquidityFarm.sol#L50)
- [IERC20(token1).transfer(stakedToken,exactAmountOut)](contracts/solidity/rewards/LiquidityFarm.sol#L55)
- [IERC20(token0).transfer(stakedToken,exactAmountOut)](contracts/solidity/rewards/LiquidityFarm.sol#L58)
- [IERC20(token).transferFrom(msg.sender,stakedToken,restAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L61)
- [liquidityAmount = pair.mint(address(this))](contracts/solidity/rewards/LiquidityFarm.sol#L64)
State variables written after the call(s):
- [_mint(msg.sender,farmTokenAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L67)
 - [_balances[account] += amount](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L263)
- [_mint(msg.sender,farmTokenAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L67)
 - [_totalSupply += amount](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L262)

contracts/solidity/rewards/LiquidityFarm.sol#L24-L73

reentrancy-events

Reentrancy vulnerabilities

Configuration

- Check: `reentrancy-events`
- Severity: `Low`
- Confidence: `Medium`

Description

Detection of the reentrancy bug. Only report reentrancies leading to out-of-order events.

Exploit Scenario:

```
function bug(Called d){
    counter += 1;
    d.f();
    emit Counter(counter);
}
```

If `d.()` re-enters, the `Counter` events will be shown in an incorrect order, which might lead to issues for third parties.

Recommendation

Apply the `check-effects-interactions` pattern.

- ID-117

Reentrancy in [BSCRewardsController.distributeEth()](contracts/solidity/bsc/BSCRewardsController.sol#L36-L46):
 External calls:
 - [UniswapV2Router(address(this)).swapExactETHForTokens{value: eth}(0x1,path,rewardsFarm,type()(uint256).max)]
(contracts/solidity/bsc/BSCRewardsController.sol#L43)
 Event emitted after the call(s):
 - [RewardsDistributed(block.number,eth,address(0x0))](contracts/solidity/bsc/BSCRewardsController.sol#L44)

contracts/solidity/bsc/BSCRewardsController.sol#L36-L46

- ID-118

Reentrancy in [SmartyProcessingPayments.payERC(address,address[],uint256,uint256,uint256)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L59-L64):
 External calls:
 - [internalPay(stealth,path,amountInMax,int256(amountOut),msg.sender,0,deadline,router,true)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L62)
  - [financeRouter.pay(targetToken,sender,destination,amountOut)](contracts/solidity/libraries/SmartyTransfer.sol#L15)
  - [financeRouter.swapTokensForExactTokens(amountInMax,amountOut,path,sender,destination,deadline)]
(contracts/solidity/libraries/SmartyTransfer.sol#L17)
  - [IInvoicesStorageManager(invoicesStorage).storeInvoice(key,amountOut)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L243)
  - [SmartyTransfer.transferFunds(path,customer,invoicesStorage,amountInMax,uint256(amountOut),deadline,financeRouter)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L247)
 Event emitted after the call(s):
 - [InvoicePaid(stealth,path[path.length - 1],uint256(amountOut))]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L63)

contracts/solidity/payments/SmartyProcessingPayments.sol#L59-L64

- ID-119

Reentrancy in [MultichainRewardsController.distribute(address[],address)]
(contracts/solidity/multichain/MultichainRewardsController.sol#L101-L113):
 External calls:
 - [paid = swapExactTokensForTokens(tokenBalance,0x1,path,address(this),address(this),type()(uint256).max)]
(contracts/solidity/multichain/MultichainRewardsController.sol#L108)
  - [require(bool,string)(IERC20(token).transfer(to,value),NCPS-06.006:FUND_POOL_FAILED)]
(contracts/solidity/rewards/BaseRewardsController.sol#L48)
  - [IUniswapV2Pair(UniswapV2Library.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0))]
(contracts/solidity/uniswap/UniswapV2Router.sol#L56-L58)
 - [MultichainRouter(multichainRouter).anySwapOutUnderlying(wrappedToken,destRewardsController,paid,destChainId)]
(contracts/solidity/multichain/MultichainRewardsController.sol#L110)
 Event emitted after the call(s):
 - [RewardsDistributed(block.number,tokenBalance,path[0])]
(contracts/solidity/multichain/MultichainRewardsController.sol#L111)

contracts/solidity/multichain/MultichainRewardsController.sol#L101-L113

- ID-120

Reentrancy in [SmartyProcessingPayments.multiDelegatePay(Structs.DelegatePayment[])]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L213-L228):
 External calls:
 - [delegatePay(payments[i],true,digest)](contracts/solidity/payments/SmartyProcessingPayments.sol#L225)
  - [financeRouter.pay(targetToken,sender,destination,amountOut)](contracts/solidity/libraries/SmartyTransfer.sol#L15)
  - [financeRouter.swapTokensForExactTokens(amountInMax,amountOut,path,sender,destination,deadline)]
(contracts/solidity/libraries/SmartyTransfer.sol#L17)
  - [IInvoicesStorageManager(invoicesStorage).storeInvoice(key,amountOut)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L243)
  - [SmartyTransfer.transferFunds(path,customer,invoicesStorage,amountInMax,uint256(amountOut),deadline,financeRouter)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L247)
 Event emitted after the call(s):
 - [InvoicePaid(payments[i].stealth,payments[i].path[payments[i].path.length - 1],uint256(payments[i].amountOut))]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L226)

contracts/solidity/payments/SmartyProcessingPayments.sol#L213-L228

- ID-121

Reentrancy in [BaseFarm.burn(uint256,address)](contracts/solidity/rewards/BaseFarm.sol#L47-L53):
 External calls:
 - [require(bool,string)(IERC20(stakedToken).transfer(to,stakedTokenAmount),NCPS-05.024:TRANSFER_FAILED)]
(contracts/solidity/rewards/BaseFarm.sol#L51)
 Event emitted after the call(s):
 - [TokenWithdrawn(block.number,stakedToken,to,stakedTokenAmount)](contracts/solidity/rewards/BaseFarm.sol#L52)

contracts/solidity/rewards/BaseFarm.sol#L47-L53

- ID-122

Reentrancy in [LiquidityFarm.stakeToken(address,uint256,bool)](contracts/solidity/rewards/LiquidityFarm.sol#L24-L73):
 External calls:
 - [IERC20(token).transferFrom(msg.sender,stakedToken,swapAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L49)
 - [pair.swap(amount0Out,amount1Out,address(this),new bytes(0))](contracts/solidity/rewards/LiquidityFarm.sol#L50)
 - [IERC20(token1).transfer(stakedToken,exactAmountOut)](contracts/solidity/rewards/LiquidityFarm.sol#L55)
 - [IERC20(token0).transfer(stakedToken,exactAmountOut)](contracts/solidity/rewards/LiquidityFarm.sol#L58)
 - [IERC20(token).transferFrom(msg.sender,stakedToken,restAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L61)
 - [liquidityAmount = pair.mint(address(this))](contracts/solidity/rewards/LiquidityFarm.sol#L64)
 Event emitted after the call(s):
 - [RewardsDistributed(block.number,swapAmount + restAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L70)
 - [TokenStaked(block.number,stakedToken,msg.sender,liquidityAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L68)
 - [Transfer(address(0),account,amount)](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L264)
  - [_mint(msg.sender,farmTokenAmount)](contracts/solidity/rewards/LiquidityFarm.sol#L67)

contracts/solidity/rewards/LiquidityFarm.sol#L24-L73

- ID-123

Reentrancy in [BaseFarm.withdraw(uint256,address)](contracts/solidity/rewards/BaseFarm.sol#L32-L45):
External calls:
- [require(bool,string)(IERC20(stakedToken).transfer(to,stakedTokenAmount),NCPS-05.024:TRANSFER_FAILED)]
(contracts/solidity/rewards/BaseFarm.sol#L42)
Event emitted after the call(s):
- [TokenWithdrawn(block.number,stakedToken,to,stakedTokenAmount)](contracts/solidity/rewards/BaseFarm.sol#L43)

contracts/solidity/rewards/BaseFarm.sol#L32-L45

- ID-124

Reentrancy in [BaseFarm.stake(uint256)](contracts/solidity/rewards/BaseFarm.sol#L16-L30):
External calls:
- [require(bool,string)(IERC20(stakedToken).transferFrom(msg.sender,address(this),amount),NCPS-05.013:TRANSFER_FAILED)]
(contracts/solidity/rewards/BaseFarm.sol#L23)
Event emitted after the call(s):
- [TokenStaked(block.number,stakedToken,msg.sender,amount)](contracts/solidity/rewards/BaseFarm.sol#L29)
- [Transfer(address(0),account,amount)](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L264)
- [_mint(msg.sender,farmTokenAmount)](contracts/solidity/rewards/BaseFarm.sol#L27)

contracts/solidity/rewards/BaseFarm.sol#L16-L30

- ID-125

Reentrancy in [MultichainRewardsController.distribute(address[],uint256)]
(contracts/solidity/multichain/MultichainRewardsController.sol#L31-L38):
External calls:
- [paid = swapExactTokensForTokens(amount,0x1,path,address(this),address(this),type()(uint256).max)]
(contracts/solidity/multichain/MultichainRewardsController.sol#L34)
- [require(bool,string)(IERC20(token).transfer(to,value),NCPS-06.006:FUND_POOL_FAILED)]
(contracts/solidity/rewards/BaseRewardsController.sol#L48)
- [IUniswapV2Pair(UniswapV2Library.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0))]
(contracts/solidity/uniswap/UniswapV2Router.sol#L56-L58)
- [MultichainRouter(multichainRouter).anySwapOutUnderlying(wrappedToken,destRewardsController,paid,destChainId)]
(contracts/solidity/multichain/MultichainRewardsController.sol#L36)
Event emitted after the call(s):
- [RewardsDistributed(block.number,amount,path[0])](contracts/solidity/multichain/MultichainRewardsController.sol#L37)

contracts/solidity/multichain/MultichainRewardsController.sol#L31-L38

- ID-126

Reentrancy in [SmartyProcessingWithdrawals.withdraw(Structs.WithdrawalInput,Structs.WithdrawalOutput)]
(contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139):
External calls:
- [invoicesManager.storeInvoice(key,- 1)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L103)
-
[withdraw(output.tokenOutputs[slaveIndex].token,output.tokenOutputs[slaveIndex].singleOutputs[k].destination,amount,out
put.tokenOutputs[slaveIndex].singleOutputs[k].crossChainData)]
(contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L120-L122)
- [SmartyTransfer.withdrawFunds(token,destination,amount,crossChainData)]
(contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L142)
- [require(bool,string)(IERC20(token).transfer(destination,amount),NCPS-04.011:TRANSFER_FAILED)]
(contracts/solidity/libraries/SmartyTransfer.sol#L30)
- [IERC20(token).approve(multichainRouter,amount)](contracts/solidity/libraries/SmartyTransfer.sol#L38)
- [MultichainRouter(multichainRouter).anySwapOutUnderlying(wrappedToken,destination,amount,chainId)]
(contracts/solidity/libraries/SmartyTransfer.sol#L39)
External calls sending eth:
-
[withdraw(output.tokenOutputs[slaveIndex].token,output.tokenOutputs[slaveIndex].singleOutputs[k].destination,amount,out
put.tokenOutputs[slaveIndex].singleOutputs[k].crossChainData)]
(contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L120-L122)
- [address(destination).transfer(amount)](contracts/solidity/libraries/SmartyTransfer.sol#L28)
Event emitted after the call(s):
- [InvoiceWithdrawn(recovered)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L104)

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139

- ID-127

Reentrancy in [SmartyProcessingPayments.provideSimplePayment(address,uint256,address[],uint256,int256,uint8,bytes32,bytes32,uint256,address)](contracts/solidity/payments/SmartyProcessingPayments.sol#L130-L146):
 External calls:
 - [updateCustomerNonce(delegatePay(Structs.DelegatePayment(stealth,path,amountInMax,amountOut,Structs.Proof(v,r,s),block.timestamp,deadline),false,digest),nonce)](contracts/solidity/payments/SmartyProcessingPayments.sol#L143-L144)
  - [IPaymentsManager(invoicesStorage).updateCustomersNonce(customer,nonce)](contracts/solidity/payments/SmartyProcessingPayments.sol#L281)
  - [financeRouter.pay(targetToken,sender,destination,amountOut)](contracts/solidity/libraries/SmartyTransfer.sol#L15)
  - [financeRouter.swapTokensForExactTokens(amountInMax,amountOut,path,sender,destination,deadline)](contracts/solidity/libraries/SmartyTransfer.sol#L17)
  - [IInvoicesStorageManager(invoicesStorage).storeInvoice(key,amountOut)](contracts/solidity/payments/SmartyProcessingPayments.sol#L243)
  - [SmartyTransfer.transferFunds(path,customer,invoicesStorage,amountInMax,uint256(amountOut),deadline,financeRouter)](contracts/solidity/payments/SmartyProcessingPayments.sol#L247)
 Event emitted after the call(s):
 - [SimpleInvoicePaid(stealth,targetToken,destination,uint256(amountOut))](contracts/solidity/payments/SmartyProcessingPayments.sol#L145)

contracts/solidity/payments/SmartyProcessingPayments.sol#L130-L146

- ID-128

Reentrancy in [SmartyProcessingPayments.simplePay(address,address[],uint256,uint256,uint256,address,bytes)](contracts/solidity/payments/SmartyProcessingPayments.sol#L66-L76):
 External calls:
 - [internalPay(stealth,path,amountInMax,int256(amountOut),msg.sender,0,deadline,router,false)](contracts/solidity/payments/SmartyProcessingPayments.sol#L72)
  - [financeRouter.pay(targetToken,sender,destination,amountOut)](contracts/solidity/libraries/SmartyTransfer.sol#L15)
  - [financeRouter.swapTokensForExactTokens(amountInMax,amountOut,path,sender,destination,deadline)](contracts/solidity/libraries/SmartyTransfer.sol#L17)
  - [IInvoicesStorageManager(invoicesStorage).storeInvoice(key,amountOut)](contracts/solidity/payments/SmartyProcessingPayments.sol#L243)
  - [SmartyTransfer.transferFunds(path,customer,invoicesStorage,amountInMax,uint256(amountOut),deadline,financeRouter)](contracts/solidity/payments/SmartyProcessingPayments.sol#L247)
 - [SmartyTransfer.withdrawFunds(path[path.length - 1],merchant,amountOut,crossChainData)](contracts/solidity/payments/SmartyProcessingPayments.sol#L73)
 Event emitted after the call(s):
 - [SimpleInvoicePaid(stealth,path[path.length - 1],merchant,amountOut)](contracts/solidity/payments/SmartyProcessingPayments.sol#L75)

contracts/solidity/payments/SmartyProcessingPayments.sol#L66-L76

- ID-129

Reentrancy in [MultichainRewardsController.distributeEth()](contracts/solidity/multichain/MultichainRewardsController.sol#L59-L70):
 External calls:
 - [paid = UniswapV2Router(address(this)).swapExactETHForTokens{value: eth}(0x1,path,address(this),type()(uint256).max)](contracts/solidity/multichain/MultichainRewardsController.sol#L66)
 - [MultichainRouter(multichainRouter).anySwapOutUnderlying(wrappedToken,destRewardsController,paid,destChainId)](contracts/solidity/multichain/MultichainRewardsController.sol#L67)
 External calls sending eth:
 - [paid = UniswapV2Router(address(this)).swapExactETHForTokens{value: eth}(0x1,path,address(this),type()(uint256).max)](contracts/solidity/multichain/MultichainRewardsController.sol#L66)
 Event emitted after the call(s):
 - [RewardsDistributed(block.number,eth,address(0x0))](contracts/solidity/multichain/MultichainRewardsController.sol#L68)

contracts/solidity/multichain/MultichainRewardsController.sol#L59-L70

- ID-130

Reentrancy in [SmartyProcessingPayments.pay(address,uint256,address[],uint256)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L252-L270):
External calls:
- [payAmount = router.swapExactETHForTokens{value: msg.value}(amountOutMin,path,invoicesStorage,deadline)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L266)
- [invoiceManager.storeInvoice(key,int256(payAmount))](contracts/solidity/payments/SmartyProcessingPayments.sol#L268)
External calls sending eth:
- [payAmount = router.swapExactETHForTokens{value: msg.value}(amountOutMin,path,invoicesStorage,deadline)]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L266)
Event emitted after the call(s):
- [InvoicePaid(stealth,targetToken,payAmount)](contracts/solidity/payments/SmartyProcessingPayments.sol#L269)

contracts/solidity/payments/SmartyProcessingPayments.sol#L252-L270

- ID-131

Reentrancy in [BaseRewardsController.distribute(address[],uint256)]
(contracts/solidity/rewards/BaseRewardsController.sol#L32-L39):
External calls:
- [swapExactTokensForTokens(amount,0x1,path,address(this),rewardsFarm,type()(uint256).max)]
(contracts/solidity/rewards/BaseRewardsController.sol#L34)
- [require(bool,string)(IERC20(token).transfer(to,value),NCPS-06.006:FUND_POOL_FAILED)]
(contracts/solidity/rewards/BaseRewardsController.sol#L48)
- [IUniswapV2Pair(UniswapV2Library.pairFor(factory,input,output)).swap(amount0Out,amount1Out,to,new bytes(0))]
(contracts/solidity/uniswap/UniswapV2Router.sol#L56-L58)
- [IERC20(rewardsToken).transfer(rewardsFarm,amount)](contracts/solidity/rewards/BaseRewardsController.sol#L36)
Event emitted after the call(s):
- [RewardsDistributed(block.number,amount,path[0])](contracts/solidity/rewards/BaseRewardsController.sol#L38)

contracts/solidity/rewards/BaseRewardsController.sol#L32-L39

timestamp

Block timestamp

Configuration

- Check: `timestamp`
- Severity: `Low`
- Confidence: `Medium`

Description

Dangerous usage of `block.timestamp`. `block.timestamp` can be manipulated by miners.

Exploit Scenario:

"Bob's contract relies on `block.timestamp` for its randomness. Eve is a miner and manipulates `block.timestamp` to exploit Bob's contract.

Recommendation

Avoid relying on `block.timestamp`.

- ID-132
[VestingWallet.init(address,address,address,address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L33-L56) uses timestamp for comparisons
 Dangerous comparisons:
 - [require(bool,string)(_defrostTime >= block.timestamp,NCPS-35.007:DEFROST_TIME_TOO_LOW)]
(contracts/solidity/vesting/VestingWallet.sol#L40)

contracts/solidity/vesting/VestingWallet.sol#L33-L56

- ID-133
[BasePullPayment.init(uint256,uint256,uint256,address,bytes32,uint256)]
(contracts/solidity/payments/BasePullPayment.sol#L56-L75) uses timestamp for comparisons
 Dangerous comparisons:
 - [require(bool,string)(currentTimestamp < _lastPayment + _period,NCPS-34.035:BAD_LAST_PAYMENT)]
(contracts/solidity/payments/BasePullPayment.sol#L67)

contracts/solidity/payments/BasePullPayment.sol#L56-L75

- ID-134
[SmartyProcessingPayments.isPaid(address)](contracts/solidity/payments/SmartyProcessingPayments.sol#L272-L274) uses timestamp for comparisons
 Dangerous comparisons:
 - [IInvoicesStorageManager(invoicesStorage).getInvoiceAmount(key) != 0]
(contracts/solidity/payments/SmartyProcessingPayments.sol#L273)

contracts/solidity/payments/SmartyProcessingPayments.sol#L272-L274

- ID-135
[VestingWallet.withdraw(address,uint256)](contracts/solidity/vesting/VestingWallet.sol#L58-L62) uses timestamp for comparisons
 Dangerous comparisons:
 - [require(bool,string)(defrostTime <= block.timestamp,NCPS-35.010:EARLY_WITHDRAWAL)]
(contracts/solidity/vesting/VestingWallet.sol#L59)

contracts/solidity/vesting/VestingWallet.sol#L58-L62

- ID-136

[SmartyProcessingPayments.internalPay(address,address[],uint256,int256,address,uint256,uint256,IFinanceRouter,bool)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L230-L249) uses timestamp for comparisons
Dangerous comparisons:
- [require(bool,string)(stealth != address(0x0),NCPS-03.013:STEALTH_ZERO_ADDR)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L232)
- [require(bool,string)(amountInMax > 0,NCPS-03.014:NOT_POSITIVE_INPUT)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L233)
- [require(bool,string)(amountOut > 0 || amountOut == - 1,NCPS-03.015:BAD_OUTPUT_AMOUNT)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L234)
- [i < path.length](contracts/solidity/payments/SmartyProcessingPayments.sol#L235)
- [require(bool,string)(path[i] != address(0x0),NCPS-03.017:SWAP_TOKEN_ZERO_ADDR)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L236)
- [require(bool,string)(check(path[0],customer),NCPS-03.019:CUSTOMER_BLACKLISTED)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L239)
- [require(bool,string)(! isPaid(key),NCPS-03.018:DOUBLE_SPEND_PAYMENT)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L242)
- [amountOut > 0](contracts/solidity/payments/SmartyProcessingPayments.sol#L246)

contracts/solidity/payments/SmartyProcessingPayments.sol#L230-L249

- ID-137

[BasePullPayment.pullPayment()](contracts/solidity/payments/BasePullPayment.sol#L28-L40) uses timestamp for comparisons
Dangerous comparisons:
- [require(bool,string)(lastPayment + period < block.timestamp,NCPS-34.020:EARLY_PAYMENT)] (contracts/solidity/payments/BasePullPayment.sol#L37)

contracts/solidity/payments/BasePullPayment.sol#L28-L40

- ID-138

[SmartyProcessingPayments.delegatePay(Structs.DelegatePayment,bool,bytes32)] (contracts/solidity/payments/SmartyProcessingPayments.sol#L196-L210) uses timestamp for comparisons
Dangerous comparisons:
- [wallet != address(0x0)](contracts/solidity/payments/SmartyProcessingPayments.sol#L202)

contracts/solidity/payments/SmartyProcessingPayments.sol#L196-L210

assembly

Assembly usage

Configuration

- Check: `assembly`
- Severity: `Informational`
- Confidence: `High`

Description

The use of assembly is error-prone and should be avoided.

Recommendation

Do not use `evm` assembly.

- ID-139
[SmartyCryptoProcessing._delegate(address)](contracts/solidity/SmartyCryptoProcessing.sol#L60-L83) uses assembly
 - [INLINE ASM](contracts/solidity/SmartyCryptoProcessing.sol#L61-L82)

contracts/solidity/SmartyCryptoProcessing.sol#L60-L83

- ID-140
[BaseMinimalProxyFactory.proxyExist(bytes32,address)](contracts/solidity/factory/BaseMinimalProxyFactory.sol#L35-L48) uses assembly
 - [INLINE ASM](contracts/solidity/factory/BaseMinimalProxyFactory.sol#L40-L45)

contracts/solidity/factory/BaseMinimalProxyFactory.sol#L35-L48

- ID-141
[VestingWalletFactory.init(address,bytes)](contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36) uses assembly
 - [INLINE ASM](contracts/solidity/vesting/VestingWalletFactory.sol#L17-L19)
 - [INLINE ASM](contracts/solidity/vesting/VestingWalletFactory.sol#L22-L24)
 - [INLINE ASM](contracts/solidity/vesting/VestingWalletFactory.sol#L27-L29)
 - [INLINE ASM](contracts/solidity/vesting/VestingWalletFactory.sol#L32-L34)

contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36

- ID-142
[SmartyCryptoProcessingDelegate._delegate(address)](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L143-L166) uses assembly
 - [INLINE ASM](contracts/solidity/SmartyCryptoProcessingDelegate.sol#L144-L165)

contracts/solidity/SmartyCryptoProcessingDelegate.sol#L143-L166

costly-loop

Costly operations inside a loop

Configuration

- Check: `costly-loop`
- Severity: `Informational`
- Confidence: `Medium`

Description

Costly operations inside a loop might waste gas, so optimizations are justified.

Exploit Scenario:

```
contract CostlyOperationsInLoop{

    uint loop_count = 100;
    uint state_variable=0;

    function bad() external{
        for (uint i=0; i < loop_count; i++){
            state_variable++;
        }
    }

    function good() external{
      uint local_variable = state_variable;
      for (uint i=0; i < loop_count; i++){
        local_variable++;
      }
      state_variable = local_variable;
    }
}
```

Incrementing `state_variable` in a loop incurs a lot of gas because of expensive `SSTOREs`, which might lead to an `out-of-gas`.

Recommendation

Use a local variable to hold the loop computation result.

- ID-143
[SmartyProcessingManager.unregisterToken(address)](contracts/solidity/SmartyProcessingManager.sol#L45-L54) has costly operations inside a loop:
 - [availableTokens.pop()](contracts/solidity/SmartyProcessingManager.sol#L50)

contracts/solidity/SmartyProcessingManager.sol#L45-L54

low-level-calls

Low-level calls

Configuration

- Check: `low-level-calls`
- Severity: `Informational`
- Confidence: `High`

Description

The use of low-level calls is error-prone. Low-level calls do not check for code existence or call success.

Recommendation

Avoid low-level calls. Check the call success. If the call is meant for a contract, check for code existence.

- ID-144

Low level call in [VestingWallet._burnLiquidityFarmToken(uint256)](contracts/solidity/vesting/VestingWallet.sol#L88-L95):
 - [(returnData) = liquidityFarm.call(payload)](contracts/solidity/vesting/VestingWallet.sol#L90)

contracts/solidity/vesting/VestingWallet.sol#L88-L95

- ID-145

Low level call in [VestingWallet._beforeUpdateFarm(address,address)](contracts/solidity/vesting/VestingWallet.sol#L118-L131):
 - [(returnData) = oldFarm.call(payload)](contracts/solidity/vesting/VestingWallet.sol#L121)
 - [(None,returnData) = oldFarm.call(payload)](contracts/solidity/vesting/VestingWallet.sol#L125)

contracts/solidity/vesting/VestingWallet.sol#L118-L131

missing-inheritance

Missing inheritance

Configuration

- Check: `missing-inheritance`
- Severity: `Informational`
- Confidence: `High`

Description

Detect missing inheritance.

Exploit Scenario:

```
interface ISomething {
    function f1() external returns(uint);
}

contract Something {
    function f1() external returns(uint){
        return 42;
    }
}
```

`Something` should inherit from `ISomething`.

Recommendation

Inherit from the missing interface or contract.

- ID-146
[SmartyCryptoProcessing](contracts/solidity/SmartyCryptoProcessing.sol#L10-L84) should inherit from [IBeacon](node_modules/@openzeppelin/contracts/proxy/beacon/IBeacon.sol#L9-L16)

contracts/solidity/SmartyCryptoProcessing.sol#L10-L84

naming-convention

Conformance to Solidity naming conventions

Configuration

- Check: `naming-convention`
- Severity: `Informational`
- Confidence: `High`

Description

Solidity defines a naming convention that should be followed.

Rule exceptions

- Allow constant variable name/symbol/decimals to be lowercase (`ERC20`).
- Allow `_` at the beginning of the `mixed_case` match for private variables and unused parameters.

Recommendation

Follow the Solidity naming convention.

- ID-147

Parameter [MerchantWallet.init(address)._merchant](contracts/solidity/merchant/MerchantWallet.sol#L24) is not in mixedCase

contracts/solidity/merchant/MerchantWallet.sol#L24

- ID-148

Parameter [VestingWallet.init(address,address,address,address,uint256)._defrostTime]
(contracts/solidity/vesting/VestingWallet.sol#L33) is not in mixedCase

contracts/solidity/vesting/VestingWallet.sol#L33

- ID-149

Variable [UniswapV2Router.WETH](contracts/solidity/uniswap/UniswapV2Router.sol#L12) is not in mixedCase

contracts/solidity/uniswap/UniswapV2Router.sol#L12

- ID-150

Parameter [VestingWallet.init(address,address,address,address,uint256)._rewardsFarm]
(contracts/solidity/vesting/VestingWallet.sol#L33) is not in mixedCase

contracts/solidity/vesting/VestingWallet.sol#L33

- ID-151

Variable [Migrations.last_completed_migration](contracts/solidity/Migrations.sol#L8) is not in mixedCase

contracts/solidity/Migrations.sol#L8

- ID-152

Parameter [BasePullPayment.init(uint256,uint256,uint256,address,bytes32,uint256)._amount]
(contracts/solidity/payments/BasePullPayment.sol#L56) is not in mixedCase

contracts/solidity/payments/BasePullPayment.sol#L56

- ID-153

Constant [BSCRewardsController.version](contracts/solidity/bsc/BSCRewardsController.sol#L9) is not in UPPER_CASE_WITH_UNDERSCORES

contracts/solidity/bsc/BSCRewardsController.sol#L9

- ID-154

Parameter [BasePullPayment.init(uint256,uint256,uint256,address,bytes32,uint256)._discriminator] (contracts/solidity/payments/BasePullPayment.sol#L56) is not in mixedCase

contracts/solidity/payments/BasePullPayment.sol#L56

- ID-155

Parameter [BasePullPayment.init(uint256,uint256,uint256,address,bytes32,uint256)._lastPayment] (contracts/solidity/payments/BasePullPayment.sol#L56) is not in mixedCase

contracts/solidity/payments/BasePullPayment.sol#L56

- ID-156

Parameter [BasePayment.initStealth(address)._stealth](contracts/solidity/payments/BasePayment.sol#L13) is not in mixedCase

contracts/solidity/payments/BasePayment.sol#L13

- ID-157

Parameter [BasePullPayment.init(uint256,uint256,uint256,address,bytes32,uint256)._periodsCapacity] (contracts/solidity/payments/BasePullPayment.sol#L56) is not in mixedCase

contracts/solidity/payments/BasePullPayment.sol#L56

- ID-158

Parameter [VestingWallet.init(address,address,address,address,uint256)._uniswapV2Router02] (contracts/solidity/vesting/VestingWallet.sol#L33) is not in mixedCase

contracts/solidity/vesting/VestingWallet.sol#L33

- ID-159

Parameter [SimplePullPayment.initToken(address)._token](contracts/solidity/payments/simple/SimplePullPayment.sol#L20) is not in mixedCase

contracts/solidity/payments/simple/SimplePullPayment.sol#L20

- ID-160

Parameter [BaseSimplePayment.initMerchant(address)._merchant] (contracts/solidity/payments/simple/BaseSimplePayment.sol#L20) is not in mixedCase

contracts/solidity/payments/simple/BaseSimplePayment.sol#L20

- ID-161

Parameter [BasePullPayment.init(uint256,uint256,uint256,address,bytes32,uint256)._payer] (contracts/solidity/payments/BasePullPayment.sol#L56) is not in mixedCase

contracts/solidity/payments/BasePullPayment.sol#L56

- ID-162

Parameter [BaseSimplePayment.initToken(address)._token](contracts/solidity/payments/simple/BaseSimplePayment.sol#L26) is not in mixedCase

contracts/solidity/payments/simple/BaseSimplePayment.sol#L26

- ID-163

Function [IFinanceRouter.WETH()](contracts/solidity/interfaces/IFinanceRouter.sol#L17) is not in mixedCase

contracts/solidity/interfaces/IFinanceRouter.sol#L17

- ID-164

Parameter [SmartyProcessingManager.setNativePaymentsEnable(bool)._nativePaymentsEnable] (contracts/solidity/SmartyProcessingManager.sol#L90) is not in mixedCase

contracts/solidity/SmartyProcessingManager.sol#L90

- ID-165

Parameter [VestingWallet.init(address,address,address,address,uint256)._liquidityFarm] (contracts/solidity/vesting/VestingWallet.sol#L33) is not in mixedCase

contracts/solidity/vesting/VestingWallet.sol#L33

- ID-166

Parameter [CustomerWallet.initCustomer(address)._customer](contracts/solidity/CustomerWallet.sol#L59) is not in mixedCase

contracts/solidity/CustomerWallet.sol#L59

- ID-167

Parameter [BasePullPayment.init(uint256,uint256,uint256,address,bytes32,uint256)._period] (contracts/solidity/payments/BasePullPayment.sol#L56) is not in mixedCase

contracts/solidity/payments/BasePullPayment.sol#L56

- ID-168

Parameter [VestingWallet.init(address,address,address,address,uint256)._investor] (contracts/solidity/vesting/VestingWallet.sol#L33) is not in mixedCase

contracts/solidity/vesting/VestingWallet.sol#L33

- ID-169

Constant [MultichainRewardsController.version](contracts/solidity/multichain/MultichainRewardsController.sol#L15) is not in UPPER_CASE_WITH_UNDERSCORES

contracts/solidity/multichain/MultichainRewardsController.sol#L15

- ID-170

Parameter [Ownable.transferOwnership(address)._newOwner](contracts/solidity/libraries/Ownable.sol#L20) is not in mixedCase

contracts/solidity/libraries/Ownable.sol#L20

- ID-171

Constant [SmartyProcessingManager.feeMaximumFrontier](contracts/solidity/SmartyProcessingManager.sol#L34) is not in UPPER_CASE_WITH_UNDERSCORES

contracts/solidity/SmartyProcessingManager.sol#L34

- ID-172

Parameter [BasePayment.initToken(address)._token](contracts/solidity/payments/BasePayment.sol#L19) is not in mixedCase

contracts/solidity/payments/BasePayment.sol#L19

- ID-173

Parameter [PullPayment.initToken(address)._token](contracts/solidity/payments/PullPayment.sol#L15) is not in mixedCase

contracts/solidity/payments/PullPayment.sol#L15

reentrancy-unlimited-gas

Reentrancy vulnerabilities

Configuration

- Check: `reentrancy-unlimited-gas`
- Severity: `Informational`
- Confidence: `Medium`

Description

Detection of the reentrancy bug. Only report reentrancy that is based on `transfer` or `send`.

Exploit Scenario:

```
    function callme(){
        msg.sender.transfer(balances[msg.sender]):
        balances[msg.sender] = 0;
    }
```

`send` and `transfer` do not protect from reentrancies in case of gas price changes.

Recommendation

Apply the `check-effects-interactions` pattern.

- ID-174

Reentrancy in [SmartyProcessingWithdrawals.withdraw(Structs.WithdrawalInput,Structs.WithdrawalOutput)]
(contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139):
 External calls:
 - [withdraw(output.tokenOutputs[slaveIndex].token,output.tokenOutputs[slaveIndex].singleOutputs[k].destination,amount,
output.tokenOutputs[slaveIndex].singleOutputs[k].crossChainData)]
(contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L120-L122)
  - [address(destination).transfer(amount)](contracts/solidity/libraries/SmartyTransfer.sol#L28)
 Event emitted after the call(s):
 - [InvoiceWithdrawn(recovered)](contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L104)

contracts/solidity/withdrawals/SmartyProcessingWithdrawals.sol#L59-L139

- ID-175

Reentrancy in [WETH.withdraw(uint256)](contracts/solidity/ERC20/WETH.sol#L20-L25):
 External calls:
 - [address(msg.sender).transfer(wad)](contracts/solidity/ERC20/WETH.sol#L23)
 Event emitted after the call(s):
 - [Withdrawal(msg.sender,wad)](contracts/solidity/ERC20/WETH.sol#L24)

contracts/solidity/ERC20/WETH.sol#L20-L25

similar-names

Variable names too similar

Configuration

- Check: `similar-names`
- Severity: `Informational`
- Confidence: `Medium`

Description

Detect variables with names that are too similar.

Exploit Scenario:

Bob uses several variables with similar names. As a result, his code is difficult to review.

Recommendation

Prevent variables from having similar names.

- ID-176

Variable [BaseFarm.stakedBalance(address).farmBalance](contracts/solidity/rewards/BaseFarm.sol#L70) is too similar to [ERC20._transfer(address,address,uint256).fromBalance](node_modules/@openzeppelin/contracts/token/ERC20/ERC20.sol#L236)

contracts/solidity/rewards/BaseFarm.sol#L70

too-many-digits

Too many digits

Configuration

- Check: `too-many-digits`
- Severity: `Informational`
- Confidence: `Medium`

Description

Literals with many digits are difficult to read and review.

Exploit Scenario:

```
contract MyContract{
    uint 1_ether = 10000000000000000000;
}
```

While `1_ether` looks like `1 ether`, it is `10 ether`. As a result, it's likely to be used incorrectly.

Recommendation

Use:

- Ether suffix,
- Time suffix, or
- The scientific notation

- ID-177
[VestingWalletFactory.init(address,bytes)](contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36) uses literals with too many digits:
 - [rewardsFarm = mload(uint256)(initData + 0x20 + 0x14) / 0x100000000000000000000000] (contracts/solidity/vesting/VestingWalletFactory.sol#L18)

contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36

- ID-178
[BaseMinimalProxyFactory.proxyExist(bytes32,address)](contracts/solidity/factory/BaseMinimalProxyFactory.sol#L35-L48) uses literals with too many digits:
 - [mstore(uint256,uint256)(ptr_proxyExist_asm_0 + 0x1E,0x5af43d82803e903d91602b57fd5bf3000000000000000000000000000000000000)] (contracts/solidity/factory/BaseMinimalProxyFactory.sol#L44)

contracts/solidity/factory/BaseMinimalProxyFactory.sol#L35-L48

- ID-179
[BaseMinimalProxyFactory.proxyExist(bytes32,address)](contracts/solidity/factory/BaseMinimalProxyFactory.sol#L35-L48) uses literals with too many digits:
 - [mstore(uint256,uint256)(ptr_proxyExist_asm_0,0x363d3d373d3d3d363d7300000000000000000000000000000000000000000000)] (contracts/solidity/factory/BaseMinimalProxyFactory.sol#L42)

contracts/solidity/factory/BaseMinimalProxyFactory.sol#L35-L48

- ID-180

[VestingWalletFactory.init(address,bytes)](contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36) uses literals with too many digits:
 - [liquidityFarm = mload(uint256)(initData + 0x20 + 0x28) / 0x100000000000000000000000] (contracts/solidity/vesting/VestingWalletFactory.sol#L23)

contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36

- ID-181

[VestingWalletFactory.init(address,bytes)](contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36) uses literals with too many digits:
 - [uniswapV2Router02 = mload(uint256)(initData + 0x20 + 0x3c) / 0x100000000000000000000000] (contracts/solidity/vesting/VestingWalletFactory.sol#L28)

contracts/solidity/vesting/VestingWalletFactory.sol#L12-L36

immutable-states

State variables that could be declared immutable

Configuration

- Check: `immutable-states`
- Severity: `Optimization`
- Confidence: `High`

Description

State variables that are not updated following deployment should be declared immutable to save gas.

Recommendation

Add the `immutable` attribute to state variables that never change or are set only in the constructor.

- ID-182
[Migrations.owner](contracts/solidity/Migrations.sol#L5) should be immutable

contracts/solidity/Migrations.sol#L5

- ID-183
[BadToken.allowed](contracts/solidity/test/BadToken.sol#L8) should be immutable

contracts/solidity/test/BadToken.sol#L8

- ID-184
[BaseRewardsController.rewardsFarm](contracts/solidity/rewards/BaseRewardsController.sol#L12) should be immutable

contracts/solidity/rewards/BaseRewardsController.sol#L12

- ID-185
[USDTBlackListChecker.usdt](contracts/solidity/ERC20/USDTBlackListChecker.sol#L9) should be immutable

contracts/solidity/ERC20/USDTBlackListChecker.sol#L9

- ID-186
[MultichainRewardsController.destRewardsController](contracts/solidity/multichain/MultichainRewardsController.sol#L10) should be immutable

contracts/solidity/multichain/MultichainRewardsController.sol#L10

Summary

You can see here all vulnerabilities found and fixed after the audit.

№	Vulnerability ID	Severity	Total found	Fixed
1	arbitrary-send-erc20	High	4	4
2	arbitrary-send-eth	High	4	4
3	reentrancy-eth	High	1	1
4	suicidal	High	1	1
5	unchecked-transfer	High	7	7
6	incorrect-equality	Medium	1	1
7	locked-ether	Medium	1	1
8	reentrancy-no-eth	Medium	4	4
9	uninitialized-local	Medium	16	16
10	unused-return	Medium	17	17
11	shadowing-local	Low	6	6
12	events-access	Low	1	1
13	events-maths	Low	2	2
14	missing-zero-check	Low	22	22
15	calls-loop	Low	28	28
16	reentrancy-benign	Low	2	2
17	reentrancy-events	Low	15	15
20	timestamp	Low	7	7
21	assembly	Informational	4	4
22	costly-loop	Informational	1	1
23	low-level-calls	Informational	2	2
24	missing-inheritance	Informational	1	1
25	naming-convention	Informational	27	27

№	Vulnerability ID	Severity	Total found	Fixed
26	reentrancy-unlimited-gas	Informational	2	2
27	similar-names	Informational	1	1
28	too-many-digits	Informational	5	5
29	immutable-states	Optimization	5	5
Total			187	187