QuillAudits

Audit Report
June, 2021

WSG

Contents

Scope of Audit

The scope of this audit was to analyze and document the Wall Street Games smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contracts care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

	High	Medium	Low	Informational
Open	0	3	1	1
Closed	0	0	0	0

Introduction

During the period of **June 06, 2021 to June 09, 2021** - QuillAudits Team performed a security audit for Wall Street Games smart contracts.

The code for the audit was taken from following the official link: https://bscscan.com/token/0xA58950F05FeA2277d2608748412bf9F802eA4901

Issues Found – Code Review / Manual Testing

High severity issues

No Issues found.

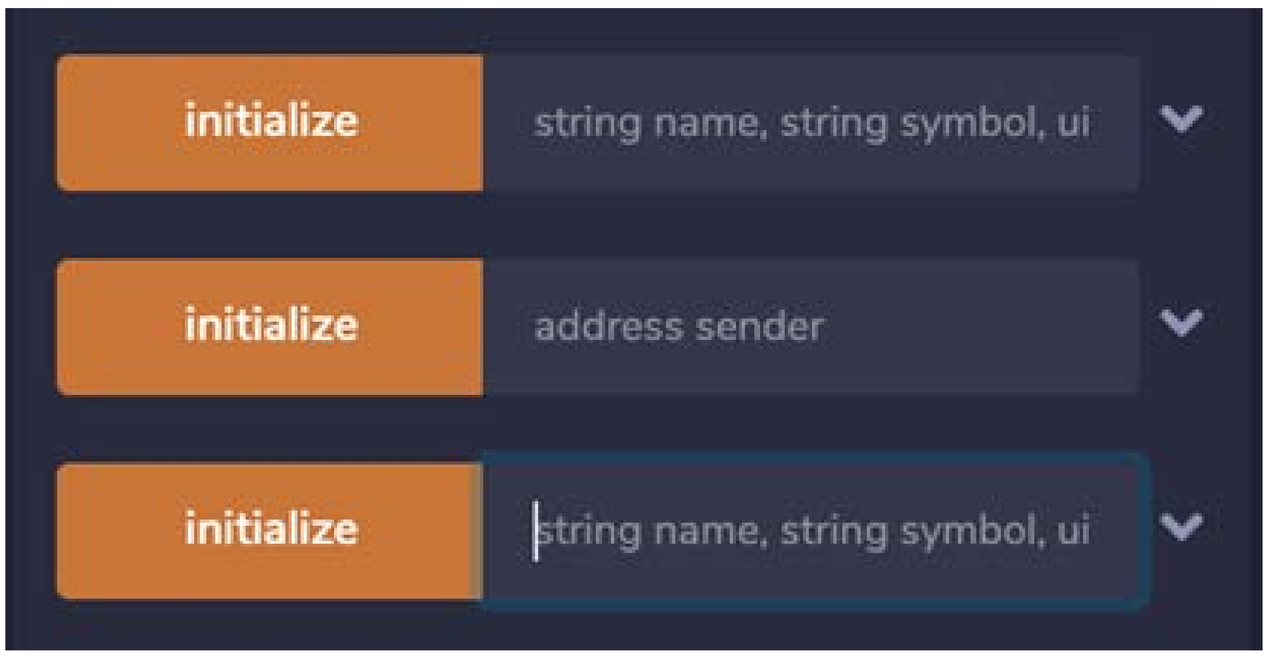
Medium severity issues

1. Duplicated initialize() functions can be called.

Line	Function Names

625	function initialize(string memory name, string memory symbol, uint8 decimals) public initializer {
	_name = name;
	_symbol = symbol;
	_decimals = decimals;
	}

	Other initializes in the following functions: Ownable ERC20Detailed ERC20Mintable ERC20Pausable



Description:
Duplicated initialize() function can be called by the owner. If the owner does not specify the number of the parameters correctly, he/she might execute the wrong function to initiate the contract.

Remediation:
Set other initialize() functions to internal.

Status: Acknowledged by the auditee.

2. Business Logic

Line	Code
1066	function() external payable {}

Description:
Contract with a payable function, but without a withdrawal capacity. Therefore, every Ether sent to this contract will be lost.

Remediation:
Remove the payable attribute or add a withdraw function.

Status: Acknowledged by the auditee.

3. Missing zero address validation

Line	Code

1068	function initialize(
	string calldata name,
	string calldata symbol,
	uint8 decimals,
	address _tokenOwner
	) external initializer {
	Ownable.initialize(msg.sender);
	
	tokenOwner = _tokenOwner;
	
	ERC20Detailed.initialize(name, symbol, decimals);
	
	ERC20Mintable.initialize(address(this));
	_removeMinter(address(this));
	_addMinter(tokenOwner);
	
	ERC20Pausable.initialize(address(this));
	_removePauser(address(this));
	_addPauser(tokenOwner);
	
	_transferOwnership(tokenOwner);
	}

Description:
Detected missing zero address validation.

Remediation:
The function should check if the address parameter is Zero.

Status: Acknowledged by the auditee.

Low level severity issues

4. transferOwnership() function is callable.

Line	Code

1029	function transferOwnership(address newOwner) public onlyOwner {
	_transferOwnership(newOwner);
	}

Description:
The transferOwnership() function is callable by the current owner. While the setOwner() function has been developed to carry out the function role.

If this function is accidentally called, the new owner must be changed to the previous owner before other functions, such as Pause, unPause, Mint, and so on, can be fully executed.

Remediation:
Set the transferOwnership() function to internal

Status: Acknowledged by the auditee.

Informational

5. Different pragma directives are used

Version used:
0.5.16
>=0.4.24<0.7.0
^0.5.0

Description:
It is detected when different Solidity versions are used.

Remediation:
Use one Solidity version for all contracts. The pragma version 0.5.16 is safe to use in this token.

Status: Acknowledged by the auditee.

Functional test

Function Names	Testing results
initialize	Passed
setOwner	Passed
transferOwnership	Failed
Paused	Passed
UnPaused	Passed
addPauser	Passed
addMinter	Passed
transfer	Passed
approve	Passed
trasferFrom	Passed

Automated Testing

Slither

```
INFO:Detectors:
ERC20._____gap (wsg.sol#571) shadows:
        - Initializable._____gap (wsg.sol#67)
ERC20Burnable._____gap (wsg.sol#603) shadows:
        - ERC20._____gap (wsg.sol#571)
        - Initializable._____gap (wsg.sol#67)
ERC20Detailed._____gap (wsg.sol#662) shadows:
        - Initializable._____gap (wsg.sol#67)
MinterRole._____gap (wsg.sol#752) shadows:
        - Initializable._____gap (wsg.sol#67)
ERC20Mintable._____gap (wsg.sol#785) shadows:
        - MinterRole._____gap (wsg.sol#752)
        - ERC20._____gap (wsg.sol#571)
        - Initializable._____gap (wsg.sol#67)
PauserRole._____gap (wsg.sol#836) shadows:
        - Initializable._____gap (wsg.sol#67)
Pausable._____gap (wsg.sol#917) shadows:
        - PauserRole._____gap (wsg.sol#836)
        - Initializable._____gap (wsg.sol#67)
ERC20Pausable._____gap (wsg.sol#960) shadows:
        - Pausable._____gap (wsg.sol#917)
        - PauserRole._____gap (wsg.sol#836)
        - ERC20._____gap (wsg.sol#571)
        - Initializable._____gap (wsg.sol#67)
Ownable._____gap (wsg.sol#1042) shadows:
        - Initializable._____gap (wsg.sol#67)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variable-shadowing
INFO:Detectors:
Contract locking ether found:
        Contract JulPadToken (wsg.sol#1056-1101) has payable functions:
         - JulPadToken.fallback() (wsg.sol#1066)
        But does not have a function to withdraw the ether
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#contracts-that-lock-ether
INFO:Detectors:
ERC20Detailed.initialize(string,string,uint8).name (wsg.sol#625) shadows:
        - ERC20Detailed.name() (wsg.sol#634-636) (function)
ERC20Detailed.initialize(string,string,uint8).symbol (wsg.sol#625) shadows:
        - ERC20Detailed.symbol() (wsg.sol#642-644) (function)
ERC20Detailed.initialize(string,string,uint8).decimals (wsg.sol#625) shadows:
        - ERC20Detailed.decimals() (wsg.sol#658-660) (function)
JulPadToken.initialize(string,string,uint8,address).name (wsg.sol#1069) shadows:
        - ERC20Detailed.name() (wsg.sol#634-636) (function)
JulPadToken.initialize(string,string,uint8,address).symbol (wsg.sol#1070) shadows:
        - ERC20Detailed.symbol() (wsg.sol#642-644) (function)
JulPadToken.initialize(string,string,uint8,address).decimals (wsg.sol#1071) shadows:
        - ERC20Detailed.decimals() (wsg.sol#658-660) (function)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#local-variable-shadowing
INFO:Detectors:
Ownable.initialize(address).sender (wsg.sol#986) lacks a zero-check on :
                - _owner = sender (wsg.sol#987)
JulPadToken.initialize(string,string,uint8,address)._tokenOwner (wsg.sol#1072) lacks a zero-check on :
                - tokenOwner = _tokenOwner (wsg.sol#1076)
JulPadToken.setOwner(address)._tokenOwner (wsg.sol#1091) lacks a zero-check on :
                - tokenOwner = _tokenOwner (wsg.sol#1095)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Initializable.isConstructor() (wsg.sol#54-64) uses assembly
        - INLINE ASM (wsg.sol#62)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
INFO:Detectors:
Different versions of Solidity is used:
        - Version used: ['0.5.16', '>=0.4.24<0.7.0', '^0.5.0']
        - >=0.4.24<0.7.0 (wsg.sol#7)
        - ^0.5.0 (wsg.sol#72)
        - ^0.5.0 (wsg.sol#103)
        - ^0.5.0 (wsg.sol#182)
        - ^0.5.0 (wsg.sol#341)
        - ^0.5.0 (wsg.sol#576)
        - ^0.5.0 (wsg.sol#608)
        - ^0.5.0 (wsg.sol#667)
        - ^0.5.0 (wsg.sol#706)
        - ^0.5.0 (wsg.sol#757)
        - ^0.5.0 (wsg.sol#790)
        - ^0.5.0 (wsg.sol#841)
        - ^0.5.0 (wsg.sol#922)
        - ^0.5.0 (wsg.sol#965)
        - 0.5.16 (wsg.sol#1047)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used
INFO:Detectors:
Context._msgData() (wsg.sol#95-98) is never used and should be removed
SafeMath.div(uint256,uint256) (wsg.sol#279-281) is never used and should be removed
SafeMath.div(uint256,uint256,string) (wsg.sol#296-303) is never used and should be removed
SafeMath.mod(uint256,uint256) (wsg.sol#316-318) is never used and should be removed
```

```
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Pragma version>=0.4.24<0.7.0 (wsg.sol#7) allows old versions
Pragma version^0.5.0 (wsg.sol#72) allows old versions
Pragma version^0.5.0 (wsg.sol#103) allows old versions
Pragma version^0.5.0 (wsg.sol#182) allows old versions
Pragma version^0.5.0 (wsg.sol#341) allows old versions
Pragma version^0.5.0 (wsg.sol#576) allows old versions
Pragma version^0.5.0 (wsg.sol#608) allows old versions
Pragma version^0.5.0 (wsg.sol#667) allows old versions
Pragma version^0.5.0 (wsg.sol#706) allows old versions
Pragma version^0.5.0 (wsg.sol#757) allows old versions
Pragma version^0.5.0 (wsg.sol#790) allows old versions
Pragma version^0.5.0 (wsg.sol#841) allows old versions
Pragma version^0.5.0 (wsg.sol#922) allows old versions
Pragma version^0.5.0 (wsg.sol#965) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity
INFO:Detectors:
Variable Initializable._____gap (wsg.sol#67) is not in mixedCase
Variable ERC20._____gap (wsg.sol#571) is not in mixedCase
Variable ERC20Burnable._____gap (wsg.sol#603) is not in mixedCase
Variable ERC20Detailed._____gap (wsg.sol#662) is not in mixedCase
Variable MinterRole._____gap (wsg.sol#752) is not in mixedCase
Variable ERC20Mintable._____gap (wsg.sol#785) is not in mixedCase
Variable PauserRole._____gap (wsg.sol#836) is not in mixedCase
Variable Pausable._____gap (wsg.sol#917) is not in mixedCase
Variable ERC20Pausable._____gap (wsg.sol#960) is not in mixedCase
Variable Ownable._____gap (wsg.sol#1042) is not in mixedCase
Parameter JulPadToken.initialize(string,string,uint8,address)._tokenOwner (wsg.sol#1072) is not in mixedCase
Parameter JulPadToken.setOwner(address)._tokenOwner (wsg.sol#1091) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (wsg.sol#96)" inContext (wsg.sol#85-99)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Ownable._____gap (wsg.sol#1042) is never used in JulPadToken (wsg.sol#1056-1101)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variables
INFO:Detectors:
totalSupply() should be declared external:
        - ERC20.totalSupply() (wsg.sol#383-385)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (wsg.sol#390-392)
allowance(address,address) should be declared external:
        - ERC20.allowance(address,address) (wsg.sol#410-412)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (wsg.sol#592-594)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (wsg.sol#599-601)
name() should be declared external:
        - ERC20Detailed.name() (wsg.sol#634-636)
symbol() should be declared external:
        - ERC20Detailed.symbol() (wsg.sol#642-644)
decimals() should be declared external:
        - ERC20Detailed.decimals() (wsg.sol#658-660)
addMinter(address) should be declared external:
        - MinterRole.addMinter(address) (wsg.sol#734-736)
renounceMinter() should be declared external:
        - MinterRole.renounceMinter() (wsg.sol#738-740)
mint(address,uint256) should be declared external:
        - ERC20Mintable.mint(address,uint256) (wsg.sol#780-783)
addPauser(address) should be declared external:
        - PauserRole.addPauser(address) (wsg.sol#818-820)
renouncePauser() should be declared external:
        - PauserRole.renouncePauser() (wsg.sol#822-824)
paused() should be declared external:
        - Pausable.paused() (wsg.sol#881-883)
pause() should be declared external:
        - Pausable.pause() (wsg.sol#904-907)
unpause() should be declared external:
        - Pausable.unpause() (wsg.sol#912-915)
owner() should be declared external:
        - Ownable.owner() (wsg.sol#994-996)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (wsg.sol#1020-1023)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (wsg.sol#1029-1031)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:wsg.sol analyzed (15 contracts with 75 detectors), 74 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Mythril

```
enderphan@enderphan WSG % myth a wsg.sol

==== Integer Underflow ====
SWC ID: 101
Severity: High
Contract: JulPadToken
Function name: name()
PC address: 3518
Estimated Gas Usage: 2643 - 4762
The binary subtraction can underflow.
The operands of the subtraction operation are not sufficiently constrained. The subtraction could therefore result in an integer underflow. Prevent the underflow by checking i
nputs or ensure sure that the underflow is caught by an assertion.
--------------------
In file: wsg.sol:635

return _name

--------------------
Initial State:

Account: [CREATOR], balance: 0x2060003041567fa6, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x2, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x203e00003603d0143, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [SOMEGUY], function: initialize(string,string,uint8), txdata: 0x1624f6c6[illegible]
[illegible], value: 0x0
Caller: [ATTACKER], function: name(), txdata: 0x06fdde03, value: 0x0

==== Integer Underflow ====
SWC ID: 101
Severity: High
Contract: JulPadToken
Function name: name()
PC address: 3570
Estimated Gas Usage: 2643 - 4762
```

```
The binary subtraction can underflow.
The operands of the subtraction operation are not sufficiently constrained. The subtraction could therefore result in an integer underflow. Prevent the underflow by checking i
nputs or ensure sure that the underflow is caught by an assertion.
--------------------
In file: wsg.sol:635

return _name

--------------------
Initial State:

Account: [CREATOR], balance: 0x2060003041567fa6, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x2, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x203e00003603d0143, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [SOMEGUY], function: initialize(string,string,uint8), txdata: 0x1624f6c6[illegible]
[illegible], value: 0x0
Caller: [ATTACKER], function: name(), txdata: 0x06fdde03, value: 0x0

==== Potential denial-of-service if block gas limit is reached ====
SWC ID: 128
Severity: Low
Contract: ERC20Burnable
Function name: burnFrom(address,uint256)
PC address: 4156
Estimated Gas Usage: 14471 - 61795
Potential denial-of-service if block gas limit is reached.
A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds the block gas limit.
--------------------
In file: wsg.sol:568

_allowances[account][_msgSender()].sub(amount, "ERC20: burn amount exceeds allowance")

--------------------
Initial State:

Account: [CREATOR], balance: 0x400c0731180089000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x2, nonce:0, storage:{}
```

```
Account: [SOMEGUY], balance: 0x20000000000050100, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [ATTACKER], function: transferFrom(address,address,uint256), txdata: 0x23b872dd[illegible]
[illegible], value: 0x0
Caller: [CREATOR], function: burnFrom(address,uint256), txdata: 0x79cc6790[illegible]
[illegible], value: 0x0

==== Integer Underflow ====
SWC ID: 101
Severity: High
Contract: JulPadToken
Function name: symbol()
PC address: 6420
Estimated Gas Usage: 2685 - 4804
The binary subtraction can underflow.
The operands of the subtraction operation are not sufficiently constrained. The subtraction could therefore result in an integer underflow. Prevent the underflow by checking i
nputs or ensure sure that the underflow is caught by an assertion.
--------------------
In file: wsg.sol:643

return _symbol

--------------------
Initial State:

Account: [CREATOR], balance: 0x40000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0xf07effffc0001dff, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [SOMEGUY], function: initialize(string,string,uint8), txdata: 0x1624f6c6[illegible]
[illegible], value: 0x0
Caller: [CREATOR], function: symbol(), txdata: 0x95d89b41, value: 0x0

==== Integer Underflow ====
SWC ID: 101
Severity: High
```

```
Severity: High
Contract: JulPadToken
Function name: symbol()
PC address: 6472
Estimated Gas Usage: 2685 - 4804
The binary subtraction can underflow.
The operands of the subtraction operation are not sufficiently constrained. The subtraction could therefore result in an integer underflow. Prevent the underflow by checking i
nputs or ensure sure that the underflow is caught by an assertion.
--------------------
In file: wsg.sol:643

return _symbol

--------------------
Initial State:

Account: [CREATOR], balance: 0x40000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0xf07effffc0001dff, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [SOMEGUY], function: initialize(string,string,uint8), txdata: 0x1624f6c6[illegible]
[illegible], value: 0x0
Caller: [CREATOR], function: symbol(), txdata: 0x95d89b41, value: 0x0

==== Potential denial-of-service if block gas limit is reached ====
SWC ID: 128
Severity: Low
Contract: JulPadToken
Function name: burnFrom(address,uint256)
PC address: 10847
Estimated Gas Usage: 14517 - 61841
Potential denial-of-service if block gas limit is reached.
A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds the block gas limit.
--------------------
In file: wsg.sol:568

_allowances[account][_msgSender()].sub(amount, "ERC20: burn amount exceeds allowance")

--------------------
```

```
_allowances[account][_msgSender()].sub(amount, "ERC20: burn amount exceeds allowance")

--------------------
Initial State:

Account: [CREATOR], balance: 0x40000000000000, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x0, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x2022a084b4c, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [CREATOR], function: transferFrom(address,address,uint256), txdata: 0x23b872dd[illegible]
[illegible], value: 0x0
Caller: [SOMEGUY], function: burnFrom(address,uint256), txdata: 0x79cc6790[illegible]
[illegible], value: 0x0

==== Potential denial-of-service if block gas limit is reached ====
SWC ID: 128
Severity: Low
Contract: JulPadToken
Function name: initialize(string,string,uint8,address)
PC address: 14575
Estimated Gas Usage: 45634 - 196150
Potential denial-of-service if block gas limit is reached.
A storage modification is executed in a loop. Be aware that the transaction may fail to execute if the loop is unbounded and the necessary gas exceeds the block gas limit. Thi
s issue is reported for internal compiler generated code.
--------------------
Initial State:

Account: [CREATOR], balance: 0x22a957c49605bea3, nonce:0, storage:{}
Account: [ATTACKER], balance: 0x1, nonce:0, storage:{}
Account: [SOMEGUY], balance: 0x33e0100240044004, nonce:0, storage:{}

Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [ATTACKER], function: initialize(string,string,uint8,address), txdata: 0xde7ea79d[illegible]
[illegible], val
ue: 0x0
```

Theo

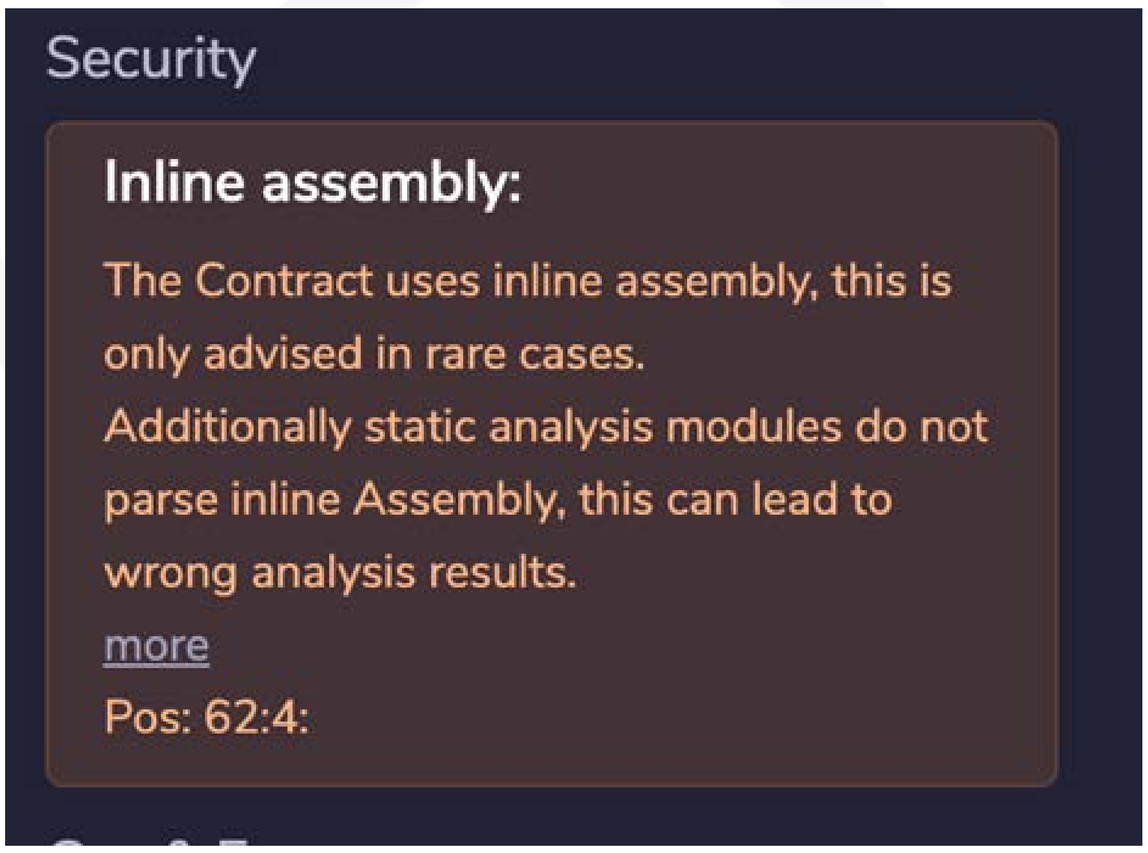
```
enderphan@enderphan ~ % theo --rpc-http http://127.0.0.1:7545
The account's private key (input hidden)
>
Contract to interact with
> 0x0CCBda68dEF63D15983F0256cBD5507Ac1FeB535
Scanning for exploits in contract: 0x0CCBda68dEF63D15983F0256cBD5507Ac1FeB535
Connecting to HTTP: http://127.0.0.1:7545.
No exploits found. You're going to need to load some exploits.

Tools available in the console:
- `exploits` is an array of loaded exploits found by Mythril or read from a file
- `w3` an initialized instance of web3py for the provided HTTP RPC endpoint
- `dump()` writing a json representation of an object to a local file

Check the readme for more info:
https://github.com/cleanunicorn/theo

Theo version v0.8.2.
```

Solhint Linter

```
wsg.sol:7:1: Error: Compiler version
>=0.4.24 <0.7.0 does not satisfy the r
semver requirement

wsg.sol:62:5: Error: Avoid to use inline
assembly. It is acceptable only in rare cases

wsg.sol:72:1: Error: Compiler version ^0.5.0
does not satisfy the r semver requirement

wsg.sol:103:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:182:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:341:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:576:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement
```

```
wsg.sol:706:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:757:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:790:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:841:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:922:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:965:1: Error: Compiler version
^0.5.0 does not satisfy the r semver
requirement

wsg.sol:1047:1: Error: Compiler version
0.5.16 does not satisfy the r semver
requirement
```

Solidity Static Analysis



Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Re-entrancy or Back-Door Entry were found in the contract.

Numerous issues of various severity levels were discovered during the audit. It is recommended to kindly go through the above-mentioned details and fix the code accordingly.

Disclaimer

Quillhash audit is not a security warranty, investment advice, or an endorsement of the Wall Street Games platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Wall Street Games Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

WSG

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com