HACKEN

SMART CONTRACT SECONDARY CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Gekkoin
Platform: Ethereum
Language: Solidity
Initial Report Date: September 24, 2019
Secondary Report Date: October 21, 2019

Document

Name	Smart Contract Secondary Code Review and Security Analysis Report for Gekkoin
Platform	Ethereum / Solidity
Link	https://github.com/gekkoin/gekkoin-tokens
Branch	audit_fixes
Commit	b06ffa0bc50c2871951697609ab22da05d5b61f5
Initial Report Date	24.09.2019
Final Report Date	21.10.2019

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Gekkoin (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of Customer`s smart contract and its code review conducted between September 20th, 2019 - September 24th, 2019; the secondary review of the code conducted between October 18th, 2019 - October 21st, 2019

Scope

The scope of the project is smart contracts, which can be found by link below:
https://github.com/gekkoin/gekkoin-tokens
We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered (the full list includes them but is not limited to them):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, Customer`s smart contract is well-secured.

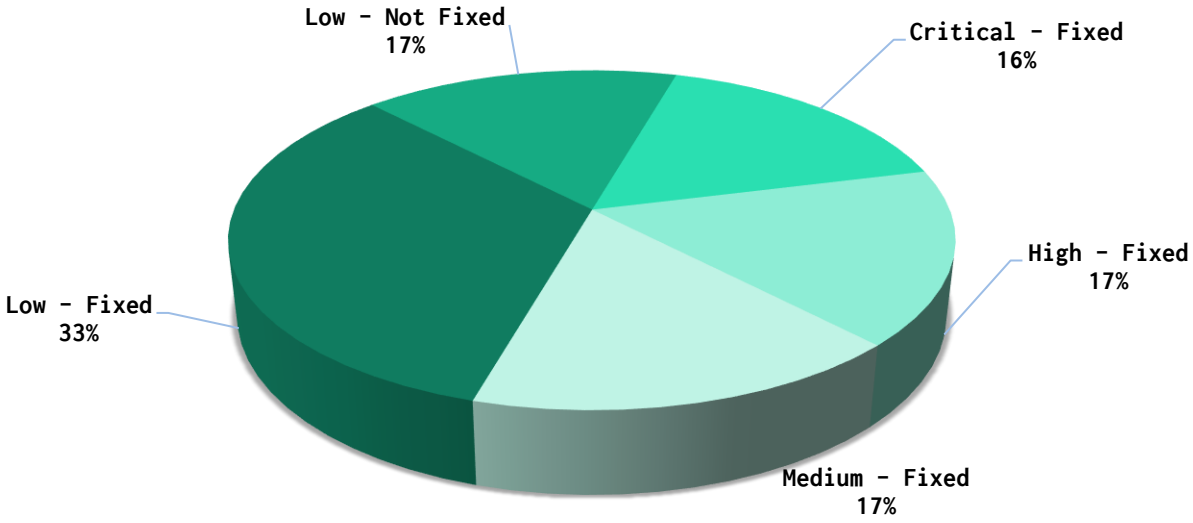
Insecure	Poor secured	Secured	Well-secured

You are here

Our team performed analysis of code functionality, manual audit and automated checks with Mythril, Slither and remix IDE (see Appendix A pic 1-2). All issues found during automated analysis were manually reviewed and applicable vulnerabilities are presented in Audit overview section. General overview is presented in AS-IS section and all found issues can be found in Audit overview section.

During the initial audit, we found 1 critical, 1 high, 1 medium and 3 low issues in smart contract. Most of them were fixed before secondary audit - only 1 low issue remained.

Graph 1. The distribution of vulnerabilities for secondary audit



Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to tokens lose etc.
High	High-level vulnerabilities are difficult to exploit; however, they also have significant impact on smart contract execution, e.g. public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they can't lead to tokens lose
Low	Low-level vulnerabilities are mostly related to outdated, unused etc. code snippets, that can't have significant impact on execution
Lowest / Code Style / Best Practice	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

GekkoinEURToken consists of the next smart contracts and libraries: **ERC20Basic**, **ERC20**, **SafeMath**, **BasicToken**, **StandardToken**, **Ownable**, **MintableToken**, **BurnableToken**, **GekkoinEURToken**.

ERC20Basic, **ERC20**, **SafeMath**, **BasicToken**, **StandardToken**, **Ownable**, **MintableToken**, **BurnableToken** are ZeppelinSolidity contracts with several modifications that were done because of business requirements:

1. Modifier canMint and function finishMinting were removed from original OpenZeppelin implementation (parameter mintingFinished and event MintFinished as well)
2. Function renounceOwnership was removed from original OpenZeppelin implementation (with OwnershipRenounced event).

GekkoinEURToken inherits **MintableToken**, **BurnableToken** and has following public parameters:

- **name** that is set to "Gekkoin EUR Token"
- **symbol** that is set to "EURG"
- **decimals** that is set to 2

Audit overview

Critical - Fixed

1. Integer overflow in burn and createTokens functions. Burn function also violates ERC20 standard. The second parameter of _burn should be _value and count.

```
function burn(uint256 _value) public {

  _burn(msg.sender, _value * (10 ** 18));

}
```

```
uint256 tokenAmount = count * (10 ** 18);
```

The issue was fixed in b06ffa0 commit.

High - Fixed

2. GekkoinEURToken owner functions can’t be called. GekkoinEUR contract will be owner of GekkoinEURToken contract after the deployment. It doesn’t implement functions to call GekkoinEURToken functions with onlyOwner modifier except for mint. It means that renounceOwnership, transferOwnership, finishMinting functions can be ever called. It’s recommended to implement functions that call functions listed above in GekkoinEUR contract. Morevover, the owner of GekkoinEUR can’t be changed.

The issue was fixed in b06ffa0 commit.

Medium - Fixed

3. Visibility for balances and totalSupply_ parameters of BasicToken are not specified and are public. ZeppelinSolidity uses internal visibility for balances and totalSupply_ in BasicToken.

The issue was fixed in b06ffa0 commit.

Low

4. Customer doesn’t have any test cases developed for the contract. The test coverage is 0% what significantly increases the risks of the logic vulnerabilities.

The contract was tested manually by internal team and security auditor.

5. Customer doesn’t have documentation or requirements for the smart contract code. It makes impossible for the auditor to find any logic issues and significantly increases the risk of such issues in the deployed code.

The issue was fixed in b06ffa0 commit.

6. Contract uses outdated version of Solidity compiler; pragma is not locked for the latest version. It’s recommended to lock the pragma to the latest stable version - 0.5.3.

The issue was fixed in b06ffa0 commit.

Lowest / Code style / Best Practice

No Lowest / Code style / Best Practice issues were found.

Conclusion

Smart contract within the scope was manually reviewed and analyzed with static analysis tools. For the contract high level description of functionality was presented in As-is overview section of the report.

Audit report contains all found security vulnerabilities and other issues in the reviewed code.

During the initial audit, security engineers found critical, high and medium issues, which could lead to DoS and other serious issues with the deployed contract. All important issues were fixed before secondary audit. Only 1 issue remained in fixed contracts that doesn’t have proven security impact.

Overall quality of reviewed contracts is high and audit team considers the smart contract as secure.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

The audit makes no statements or warranties on security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on blockchain platform. The platform, its programming language, and other software related to the smart contract can have own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

Appendix A. Automated tools reports

Pic 1. Slither automated report:

```
rdchksec@ubuntu:~/gekkoin-tokens$ slither contracts/GekkoinEURToken.sol
INFO:Detectors:
ERC20Basic.totalSupply (contracts/GekkoinEURToken.sol#9) should be declared external
BasicToken.totalSupply (contracts/GekkoinEURToken.sol#96-98) should be declared external
BasicToken.balanceOf (contracts/GekkoinEURToken.sol#120-122) should be declared external
ERC20Basic.balanceOf (contracts/GekkoinEURToken.sol#10) should be declared external
BasicToken.transfer (contracts/GekkoinEURToken.sol#105-113) should be declared external
ERC20Basic.transfer (contracts/GekkoinEURToken.sol#11) should be declared external
ERC20.allowance (contracts/GekkoinEURToken.sol#20) should be declared external
StandardToken.allowance (contracts/GekkoinEURToken.sol#184-193) should be declared external
ERC20.transferFrom (contracts/GekkoinEURToken.sol#22) should be declared external
StandardToken.transferFrom (contracts/GekkoinEURToken.sol#144-161) should be declared external
StandardToken.approve (contracts/GekkoinEURToken.sol#172-176) should be declared external
ERC20.approve (contracts/GekkoinEURToken.sol#24) should be declared external
StandardToken.increaseApproval (contracts/GekkoinEURToken.sol#204-215) should be declared external
StandardToken.decreaseApproval (contracts/GekkoinEURToken.sol#226-241) should be declared external
Ownable.transferOwnership (contracts/GekkoinEURToken.sol#280-282) should be declared external
MintableToken.mint (contracts/GekkoinEURToken.sol#314-327) should be declared external
BurnableToken.burn (contracts/GekkoinEURToken.sol#347-349) should be declared external
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-as-external
INFO:Detectors:
Parameter '_owner' of BasicToken.balanceOf (contracts/GekkoinEURToken.sol#120) is not in mixedCase
Parameter '_to' of BasicToken.transfer (contracts/GekkoinEURToken.sol#105) is not in mixedCase
Parameter '_value' of BasicToken.transfer (contracts/GekkoinEURToken.sol#105) is not in mixedCase
Parameter '_owner' of StandardToken.allowance (contracts/GekkoinEURToken.sol#185) is not in mixedCase
Parameter '_spender' of StandardToken.allowance (contracts/GekkoinEURToken.sol#186) is not in mixedCase
Parameter '_from' of StandardToken.transferFrom (contracts/GekkoinEURToken.sol#145) is not in mixedCase
Parameter '_to' of StandardToken.transferFrom (contracts/GekkoinEURToken.sol#146) is not in mixedCase
Parameter '_value' of StandardToken.transferFrom (contracts/GekkoinEURToken.sol#147) is not in mixedCase
Parameter '_spender' of StandardToken.approve (contracts/GekkoinEURToken.sol#172) is not in mixedCase
Parameter '_value' of StandardToken.approve (contracts/GekkoinEURToken.sol#172) is not in mixedCase
Parameter '_spender' of StandardToken.increaseApproval (contracts/GekkoinEURToken.sol#205) is not in mixedCase
Parameter '_addedValue' of StandardToken.increaseApproval (contracts/GekkoinEURToken.sol#206) is not in mixedCase
Parameter '_spender' of StandardToken.decreaseApproval (contracts/GekkoinEURToken.sol#227) is not in mixedCase
Parameter '_subtractedValue' of StandardToken.decreaseApproval (contracts/GekkoinEURToken.sol#228) is not in mixedCase
Parameter '_newOwner' of Ownable.transferOwnership (contracts/GekkoinEURToken.sol#280) is not in mixedCase
Function 'Ownable._transferOwnership' (contracts/GekkoinEURToken.sol#288-292) is not in mixedCase
Parameter '_newOwner' of Ownable._transferOwnership (contracts/GekkoinEURToken.sol#288) is not in mixedCase
Parameter '_to' of MintableToken.mint (contracts/GekkoinEURToken.sol#315) is not in mixedCase
Parameter '_amount' of MintableToken.mint (contracts/GekkoinEURToken.sol#316) is not in mixedCase
Parameter '_value' of BurnableToken.burn (contracts/GekkoinEURToken.sol#347) is not in mixedCase
Function 'BurnableToken._burn' (contracts/GekkoinEURToken.sol#351-360) is not in mixedCase
Parameter '_who' of BurnableToken._burn (contracts/GekkoinEURToken.sol#351) is not in mixedCase
Parameter '_value' of BurnableToken._burn (contracts/GekkoinEURToken.sol#351) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Slither:contracts/GekkoinEURToken.sol analyzed (9 contracts), 40 result(s) found
```

Pic 2. Mythril automated report:

```
rdchksec@ubuntu:~/gekkoin-tokens$ myth a contracts/GekkoinEURToken.sol
==== Exception State ====
SWC ID: 110
Severity: Low
Contract: MintableToken
Function name: mint(address,uint256)
PC address: 4994
Estimated Gas Usage: 1285 - 1710
A reachable exception has been detected.
It is possible to trigger an exception (opcode 0xfe). Exceptions can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. Note
 that explicit `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: contracts/GekkoinEURToken.sol:84

 version of St

--------------------
Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [CREATOR], function: mint(address,uint256), txdata: 0x40c10f198080808080808080808080808080808080808080808080808080808080808080808080808080808080802040a97
c3da00f814d16420d5ec319c69caa2c, value: 0x0
Caller: [CREATOR], function: mint(address,uint256), txdata: 0x40c10f197f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f7f8010001f79c
0fda3409f550a408444216a8288a698, value: 0x0

==== Exception State ====
SWC ID: 110
Severity: Low
Contract: GekkoinEURToken
Function name: mint(address,uint256)
PC address: 5638
Estimated Gas Usage: 1285 - 1710
A reachable exception has been detected.
It is possible to trigger an exception (opcode 0xfe). Exceptions can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. Note
 that explicit `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: contracts/GekkoinEURToken.sol:84

 version of St

--------------------
Transaction Sequence:

Caller: [CREATOR], data: [CONTRACT CREATION], value: 0x0
Caller: [CREATOR], function: mint(address,uint256), txdata: 0x40c10f193232323232323232323232323232323232323232323232323232323232323232838465328c6b3232dc05ae926d5764954
c8a9950481186428008e01c2c110185, value: 0x0
Caller: [CREATOR], function: mint(address,uint256), txdata: 0x40c10f194141414141414141414141414141414141414141414141414141414141414141cd14ada095a26222803c0430aa010c151
4964d51c810a24280772063d5f6fe7b, value: 0x0
```