Smart Contract Code Review and Security Analysis Report

Customer: 0xcert
Date: June 29, 18

This document contains confidential information about IT systems and intellectual properties of the customer, as well as information about potential vulnerabilities and methods of their exploitation.
This confidential information is for internal use by the customer only and shall not be disclosed to third parties.

Document:

Name:	Smart Contract Code Review and Security Analysis Report for 0xcert
Date:	29.06.2018

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by 0xcert (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 June 19th, 2018 - June 29th, 2018.

Scope

The scope of the project is 0xcert smart contracts, which can be found on github by links below:

- Zxc Token https://github.com/0xcert/ethereum-zxc/tree/master/contracts
  - Commit - 1fafdebea4ae77ebc5dac574485b8f28c3ea93f7
- Xcert https://github.com/0xcert/ethereum-xcert/tree/master/contracts
  - Commit - bea01c566327fc98fa077ba92ef593548efc7365
- Crowdsale https://github.com/0xcert/ethereum-crowdsale/tree/master/contracts
  - Commit - 3a11ff95121f6a71e0052fd95a9a23a88b622499

The full version of tested and audited contracts is: ERC165, ERC20, ERC721, ERC721Enumerable, ERC721Metadata, ERC721TokenReceiver, NFToken, NFTokenEnumerable, NFTokenMetadata, Ownable, SafeMath, SupportsInterface, Token, Xcert, RevokableXcert, PausableXcert, MutableXcert, BurnableXcert, Selector, Zxc, ZxcCrowdsale.

Migrations.sol and mocks contracts used in repositories are not in scope of the project.

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
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed

- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

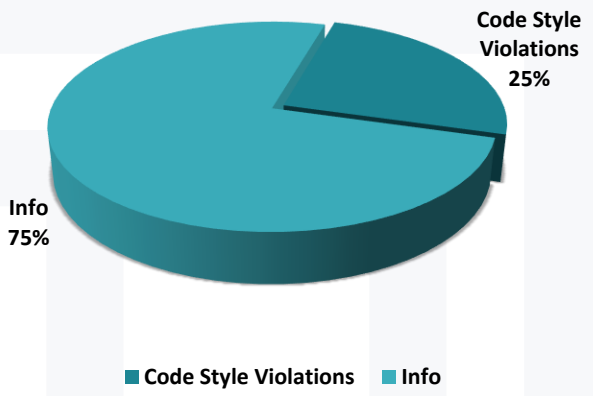
Executive Summary

According to the assessment, Customer`s smart contracts are well secured and no major fixes are required.

Our team performed analysis of code functionality, manual audit and automated checks with solc, Mythrill and remix IDE (see Appendix B pic 1-29). All found 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.

We found no security issues; however, some informational statements and code style guide violations were found. They can't have security or functional impact, but they should be analyzed by Customer and its users.

Graph 1. Vulnerabilities distribution



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 / Info	Lowest-level vulnerabilities, code style violations and info statements can't affect smart contract execution and can be ignored.

AS-IS overview

ZXC contract overview

Zxc.sol is contract, which describes standard ERC20 (inherits from Token, Ownable) token for 0xcert project.

Zxc contract constructor sets:

- tokenName to 0xcert Protocol Token
- tokenSymbol to ZXC
- tokenDecimals to 18
- tokenTotalSupply to 400000000000000000000000000
- transferEnabled to false
- balances[owner] to tokenTotalSupply

Zxc contract determines next functions:

- with validDestination modifier – guarantees that _to address is a valid destination to transfer tokens to and with onlyWhenTransferAllowed modifier – checks if tokens can be transferred
  - transfer, transferFrom functions from ERC20 standard
- external with onlyOwner modifier– allows function call to owner address:
  - enableTransfer – allows owner account to send tokens
  - burn – allows owner account to burn tokens
  - setCrowdsaleAddress – allows owner account to set crowdsaleAddress address

XCert contracts overview

XCert is ERC721 extension that describes customized non-fungible token.

XCert inherits from NFTokenEnumerable and NFTokenMetadata contracts; NFTokenEnumerable inherits from NFToken contract and ERC721Enumerable interface; NFTokenMetadata inherits from NFToken contract and ERC721Metadata interface; NFToken inherits from Ownable contract, ERC721 interface, SupportsInterface contract (which inherits from ERC165 interface).

Xcert contract determines next parameters:

- nftConventionId – ID which determines each Xcert smart contract type by its JSON convention
- idToProof mapping – maps token id to token proof
- config mapping – maps token id to token config protocol
- data mapping – maps token id to token convention data
- addressToAuthorized mapping – maps address to bool (true if address can mint new tokens)

Xcert contract determines modifier isAuthorized that requires msg.sender to be owner or addressToAuthorized[msg.sender] is true.

Xcert contract constructor sets supportedInterfaces[0x6be14f75] to true.

Xcert contract determines next functions:

- external with isAuthorized modifier:
  - mint – allows address to min new token
- external with onlyOwner modifier:
  - setAuthorizedAddress – allows owner to set addressToAuthorized[address] to true
- external view:
  - conventionId – allows to get a bytes4 of keccak256 of json schema representing 0xcert protocol convention
  - tokenProof – allows to get proof for token with given token id
  - tokenExpirationTime – allows to get expiration time for token with given token id
  - isAuthorizedAddress – allows to check whether address is authorized to mint tokens
- public view:
  - tokenDataValue – allows to get convention data for token with given token id

There are variations of Xcert in repository:

- BurnableXcert – Xcert token implementation where tokens can be burnt by contract operator
- MutableXcert – Xcert token implementation where data can be changed
- PausableXcert – XCert token implementation where tokens can be paused
- RevokableXcert – Xcert token implementation where tokens can be burnt by issuer

ZxcCrowdsale contract overview

ZxcCrowdsale contract is used for distributing ZXC tokens.

ZxcCrowdsale uses parameters Zxc token and Xcert xcertKyc to call functions in token and xcertKyc contracts. Other ZxcCrowdsale parameters are:

- startTimePresale – presale start time
- startTimeSaleWithBonus – token sale with bonus start time
- startTimeSaleNoBonus – token sale without bonus start time
- bonusPresale – percent amount for bonus on presale
- bonusSale – percent amount for bonus on crowdsale
- endTime – crowdsale end time
- minimumPresaleWeiDeposit – minimum deposit in wei for presale
- preSaleZxcCap – number of Zxc tokens available on presale
- crowdSaleZxcSupply – number of Zxc tokens available on crowdsale
- zxcSold – number of Zxc sold during the crowdsale
- wallet – address for received funds
- rate – number of tokens received per 1 wei

Xcert contract constructor sets wallet, Zxc token address, Xcert xcertKyc address, startTimePresale, startTimeSaleWithBonus, startTimeSaleNoBonus, endTime, rate, preSaleZxcCap, crowdSaleZxcSupply, bonusPresale, bonusSale, minimumPresaleWeiDeposit to values specified by ZxcCrowdsale contract owner.

ZxcCrowdsale contract determines next functions:

- external payable:
  - fallback function – calls buyTokens function
- public payable:
  - buyTokens – allows to purchase Zxc tokens
- external view:
  - hasEnded – checks whether crowdsale is ended (true if ended)
- internal view:
  - isInTimeRange – checks whether current period is in crowdsale period
  - getTokenAmount – calculates number of tokens will be received by investor for given amount of wei and given bonus percent

Audit overview

Critical

No critical vulnerabilities were found.

High

No critical vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

Lowest / Code style / Info

Informational statements

Informational statements are audit team findings that doesn't have any security issues. However, they are presented in report to clarify and outline functionality and business requirements.

1. XCert config is not changeable by contract owner or XCert owner. For instance, digital asset ExpirationTime can't be prolongated or config items can't be removed or added.
2. tokenExpirationTime is stored as first item of config array and it takes 32 bytes to be stored what is much more than needed for timestamp. It can be optmized in different ways to have some gas economy.
3. Indexes are changed after token burned/revoked. XCert contract operate with both Ids and indexes. If DApps that operate with burnable/revokable XCert and token indexes are changed after each burn and revoke, that DApps can potentially have some errors or inconveniencies in development.

Code Style Issues

Code style issues are related to Solidity Code Style Best Practices, however, they and can't have any security or functional impact.

4. Redundant code in ZxcCrowdsale.sol (lines 259-264, see Appendix A pic 1 for evidence).

```
if (now >= _startTime && now < _endTime) {
      return true;
    }
    else {
      return false;
    }

// It can be rewritten as presented below

return(now >= _startTime && now < _endTime);
```

Consider changing code as presented above.

Conclusion

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

Audit team haven't found any security issues during both manual and automated audits and the report contains informational statements and code style notice related to the reviewed code.

Overall quality of reviewed contracts is high and no fixes are required.

Disclaimers

Disclaimer

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

Technical Disclaimer

Smart contract build on the top of Ethereum blockchain means that a lot of features could be covered by tests, but Turing completeness of Solidity programming language realization leaves some space for unexpected runtime exceptions.

Appendix A. Evidences

Pic 1. Redundant code in ZxcCrowdsale.sol:

```
251   function isInTimeRange(
252     uint256 _startTime,
253     uint256 _endTime
254   )
255     internal
256     view
257     returns(bool)
258   {
259     if (now >= _startTime && now < _endTime) {
260       return true;
261     }
262     else {
263       return false;
264     }
265   }
266
```

Appendix B. Automated tools reports

Pic. 1 Solc ZxcCrowdsale automated report

```
max@max-VirtualBox:~/solidity/projects/0xcert$ solc -o . --bin --abi --overwrite ZxcCrowdsale.sol
max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 2 Mythrill ZxcCrowdsale automated report

```
max@max-VirtualBox:~/solidity/projects/0xcert$ myth -x ZxcCrowdsale.sol
==== Message call to external contract ====
Type: Informational
Contract: Unknown
Function name: fallback
PC address: 1228
This contract executes a message call to to another contract. Make sure that the called contract is trusted and does not execute user-supplied code.
--------------------
In file: ZxcCrowdsale.sol:202

xcertKyc.balanceOf(msg.sender)

--------------------

==== Multiple Calls ====
Type: Information
Contract: Unknown
Function name: fallback
PC address: 1228
Multiple sends exist in one transaction, try to isolate each external call into its own transaction. As external calls can fail accidentally or deliberately.
Consecutive calls:
Call at address: 1754
Call at address: 1539
Call at address: 1754
Call at address: 1539

--------------------
In file: ZxcCrowdsale.sol:202

xcertKyc.balanceOf(msg.sender)

--------------------

==== Message call to external contract ====
Type: Informational
Contract: Unknown
Function name: fallback
PC address: 1539
This contract executes a message call to to another contract. Make sure that the called contract is trusted and does not execute user-supplied code.
--------------------
In file: ZxcCrowdsale.sol:206

xcertKyc.tokenOfOwnerByIndex(msg.sender, balance.sub(1))

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

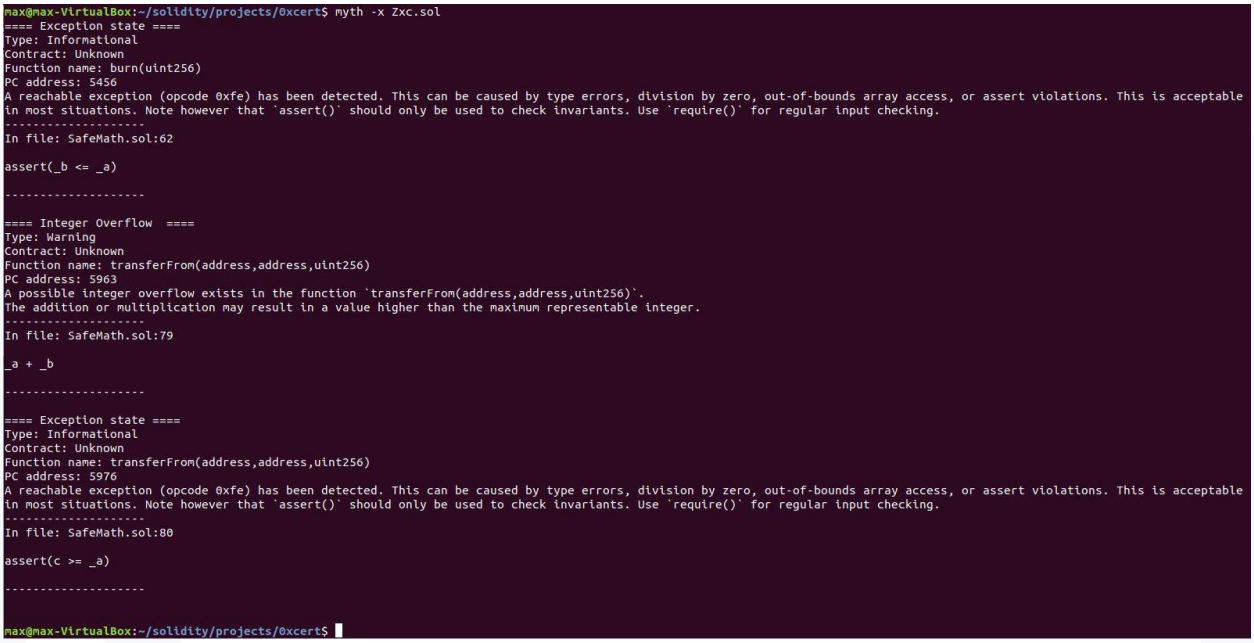
Pic. 3 Mythrill ZxcCrowdsale automated report

```
==== Multiple Calls ====
Type: Information
Contract: Unknown
Function name: fallback
PC address: 1539
Multiple sends exist in one transaction, try to isolate each external call into its own transaction. As external calls can fail accidentally or deliberately.
Consecutive calls:
Call at address: 1754

--------------------
In file: ZxcCrowdsale.sol:206

xcertKyc.tokenOfOwnerByIndex(msg.sender, balance.sub(1))

--------------------

==== Message call to external contract ====
Type: Informational
Contract: Unknown
Function name: fallback
PC address: 1754
This contract executes a message call to to another contract. Make sure that the called contract is trusted and does not execute user-supplied code.
--------------------
In file: ZxcCrowdsale.sol:207

xcertKyc.tokenDataValue(tokenId, 0)

--------------------


max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 4 Solc Zxc automated report

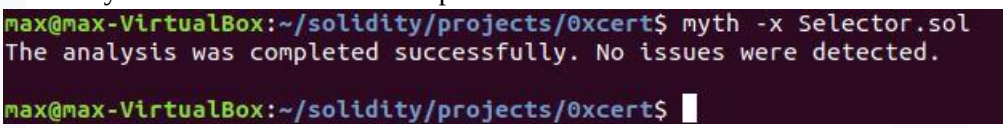
```
max@max-VirtualBox:~/solidity/projects/0xcert$ solc -o . --bin --abi --overwrite Zxc.sol
max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 5 Mythrill Zxc automated report

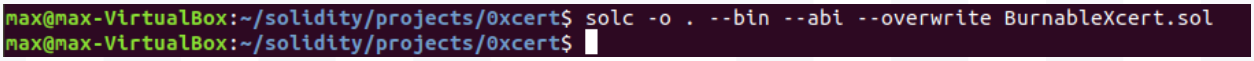


Pic. 6 Solc Selector automated report

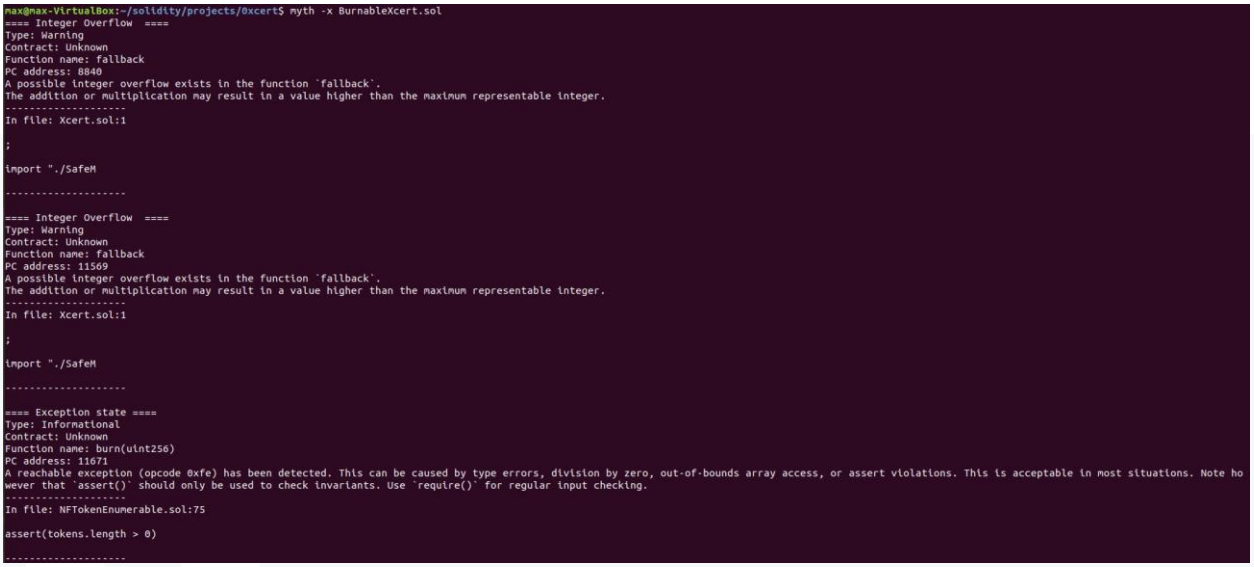
Pic. 7 Mythrill Selector automated report



Pic. 8 Solc BurnableXcert automated report

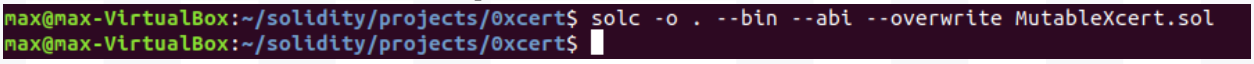


Pic. 9 Mythrill BurnableXcert automated report



Pic. 10 Mythrill BurnableXcert automated report

Pic. 11 Solc MutableXcert automated report



Pic. 12 Mythrill MutableXcert automated report

```
max@max-VirtualBox:~/solidity/projects/0xcert$ myth -x MutableXcert.sol
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: setTokenData(uint256,bytes32[])
PC address: 1584
A possible integer overflow exists in the function `setTokenData(uint256,bytes32[])`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: MutableXcert.sol:36

function setTokenData(
    uint256 _tokenId,
    bytes32[] _data
  )
    validNFToken(_tokenId)
    isAuthorized()
    external
  {
    data[_tokenId] = _data;
    emit TokenDataChange(_tokenId, _data);
  }

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: setTokenData(uint256,bytes32[])
PC address: 1590
A possible integer overflow exists in the function `setTokenData(uint256,bytes32[])`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: MutableXcert.sol:36

function setTokenData(
    uint256 _tokenId,
    bytes32[] _data
  )
    validNFToken(_tokenId)
    isAuthorized()
    external
  {
    data[_tokenId] = _data;
    emit TokenDataChange(_tokenId, _data);
  }

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

Pic. 13 Mythrill MutableXcert automated report

```
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 8796
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Xcert.sol:1

;

import "./SafeM

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 11430
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Xcert.sol:1

;

import "./SafeM

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 12281
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: SafeMath.sol:79

_a + _b

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

Pic. 14 Mythrill MutableXcert automated report

```
==== Exception state ====
Type: Informational
Contract: Unknown
Function name: fallback
PC address: 12294
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note ho
wever that `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: SafeMath.sol:80

assert(c >= _a)

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

Pic. 15 Mythrill MutableXcert automated report

```
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: setTokenData(uint256,bytes32[])
PC address: 12462
A possible integer overflow exists in the function `setTokenData(uint256,bytes32[])`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: MutableXcert.sol:8

contract MutableXcert is Xcert {

  /**
   * @dev Emits when an Token data is changed.
   * @param _id NFT that data got changed.
   * @param _data New data.
   */
  event TokenDataChange(
    uint256 indexed _id,
    bytes32[] _data
  );

  /**
   * @dev Contract constructor.
   * @notice When implementing this contract don't forget to set nftConventionId, nftName and
   * nftSymbol.
   */
  constructor()
    public
  {
    supportedInterfaces[0x59118221] = true; // MutableXcert
  }

  /**
   * @dev Modifies convention data by setting a new value for a given index field.
   * @param _tokenId Id of the NFT we want to set key value data.
   * @param _data New token data.
   */
  function setTokenData(
    uint256 _tokenId,
    bytes32[] _data
  )
    validNFToken(_tokenId)
    isAuthorized()
    external
  {
    data[_tokenId] = _data;
    emit TokenDataChange(_tokenId, _data);
  }
}

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

Pic. 16 Mythrill MutableXcert automated report

```
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: setTokenData(uint256,bytes32[])
PC address: 12464
A possible integer overflow exists in the function `setTokenData(uint256,bytes32[])`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: MutableXcert.sol:8

contract MutableXcert is Xcert {

  /**
   * @dev Emits when an Token data is changed.
   * @param _id NFT that data got changed.
   * @param _data New data.
   */
  event TokenDataChange(
    uint256 indexed _id,
    bytes32[] _data
  );

  /**
   * @dev Contract constructor.
   * @notice When implementing this contract don't forget to set nftConventionId, nftName and
   * nftSymbol.
   */
  constructor()
    public
  {
    supportedInterfaces[0x59118221] = true; // MutableXcert
  }

  /**
   * @dev Modifies convention data by setting a new value for a given index field.
   * @param _tokenId Id of the NFT we want to set key value data.
   * @param _data New token data.
   */
  function setTokenData(
    uint256 _tokenId,
    bytes32[] _data
  )
    validNFToken(_tokenId)
    isAuthorized()
    external
  {
    data[_tokenId] = _data;
    emit TokenDataChange(_tokenId, _data);
  }
}

--------------------


max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 17 Solc PausableXcert automated report

```
max@max-VirtualBox:~/solidity/projects/0xcert$ solc -o . --bin --abi --overwrite PausableXcert.sol
max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 18 Mythrill PausableXcert automated report

```
max@max-VirtualBox:~/solidity/projects/0xcert$ myth -x PausableXcert.sol
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 8676
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Xcert.sol:1

;

import "./SafeM

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 11335
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Xcert.sol:1

;

import "./SafeM

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 12186
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: SafeMath.sol:79

_a + _b

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

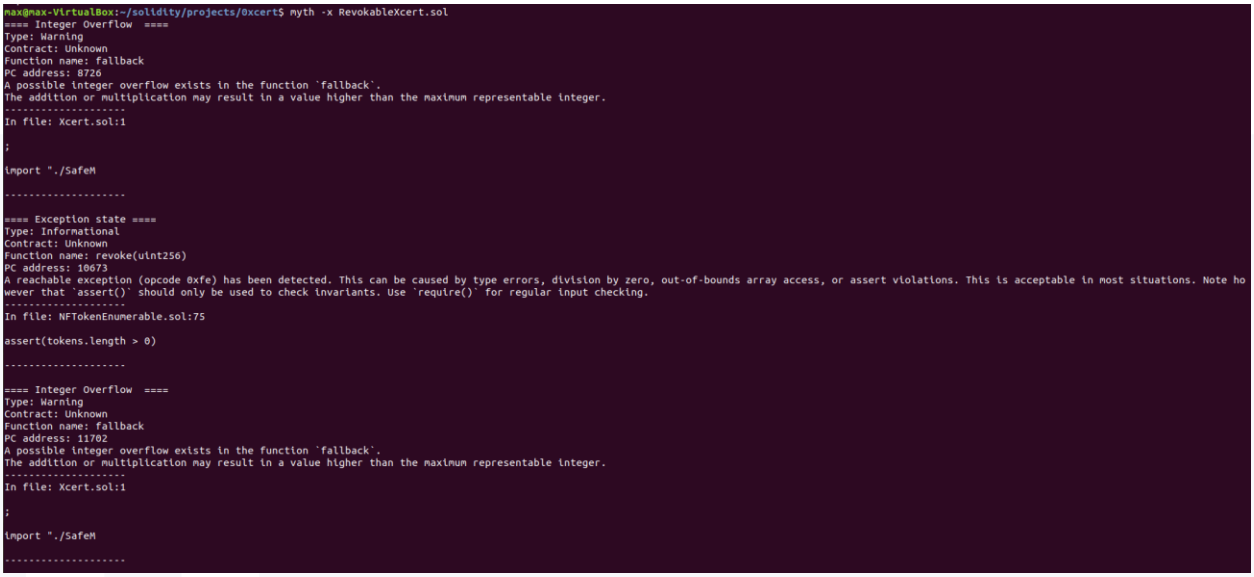
Pic. 19 Mythrill PausableXcert automated report

```
==== Exception state ====
Type: Informational
Contract: Unknown
Function name: fallback
PC address: 12199
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note however t
hat `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: SafeMath.sol:80

assert(c >= _a)

--------------------


max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 20 Solc RevokableXcert automated report

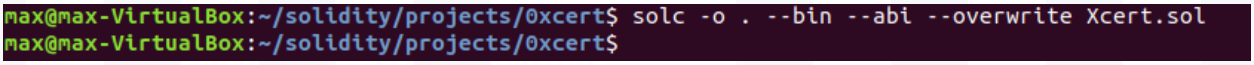
```
max@max-VirtualBox:~/solidity/projects/0xcert$ solc -o . --bin --abi --overwrite RevokableXcert.sol
max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 21 Mythrill RevokableXcert automated report



Pic. 22 Mythrill RevokableXcert automated report

Pic. 23 Solc Xcert automated report



Pic. 24 Mythrill Xcert automated report

```
max@max-VirtualBox:~/solidity/projects/0xcert$ myth -x Xcert.sol
==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 8303
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Xcert.sol:1

;

import "./SafeM

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 10937
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: Xcert.sol:1

;

import "./SafeM

--------------------

==== Integer Overflow  ====
Type: Warning
Contract: Unknown
Function name: fallback
PC address: 11788
A possible integer overflow exists in the function `fallback`.
The addition or multiplication may result in a value higher than the maximum representable integer.
--------------------
In file: SafeMath.sol:79

_a + _b

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

Pic. 25 Mythrill Xcert automated report

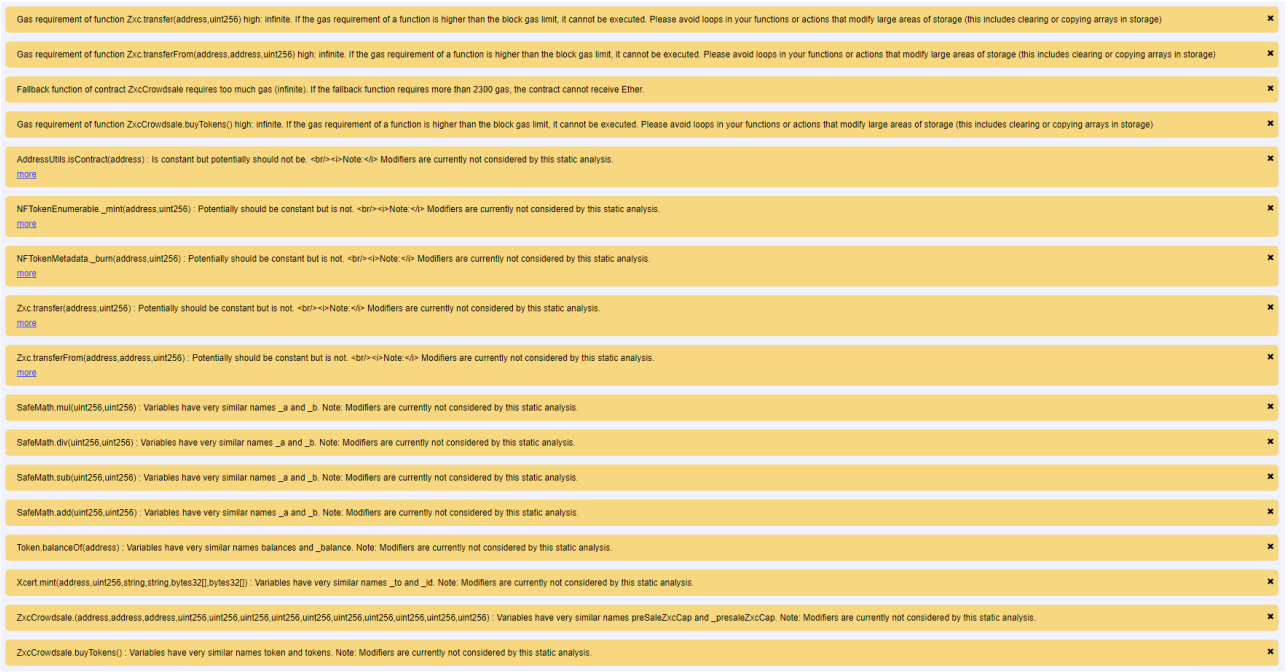
```
==== Exception state ====
Type: Informational
Contract: Unknown
Function name: fallback
PC address: 11801
A reachable exception (opcode 0xfe) has been detected. This can be caused by type errors, division by zero, out-of-bounds array access, or assert violations. This is acceptable in most situations. Note however t
hat `assert()` should only be used to check invariants. Use `require()` for regular input checking.
--------------------
In file: SafeMath.sol:80

assert(c >= _a)

--------------------

max@max-VirtualBox:~/solidity/projects/0xcert$
```

Pic. 26 List of audited contracts with Remix IDE static analysis:

Pic. 26 Remix IDE static analysis report part 1:

Pic. 27 Remix IDE static analysis report part 2:

Pic. 28 Remix IDE static analysis report part 3:



Pic. 29 Remix IDE static analysis report part 4: