its__msn itsmsn

ITISH

AUDIT COMPANY

Audit Details

Token Name
AZNT

Deployer address

TD74cpU4JfMKhuWVs6uKYXKUy8nvC6EQ95

Client contacts:

AZNT Token Team

Blockchain
TRON

Project website:

Not Provided by Token

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for you to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that you should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for you to conduct your own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

DISCLAIMER: By reading this report or any part of it, you agree to the terms of this disclaimer. If you do not agree to the terms, then please immediately cease reading this report, and delete and destroy any and all copies of this report downloaded and/or printed by you. This report is provided for information purposes only and on a non-reliance basis, and does not constitute investment advice. No one shall have any right to rely on the report or its contents, and Itish and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives) (Itish) owe no duty of care towards you or any other person, nor does Itish make any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and Itish hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effect in relation to the report. Except and only to the extent that it is prohibited by law, Itish hereby excludes all liability and responsibility, and neither you nor any other person shall have any claim against Itish, for any amount or kind of loss or damage that may result to you or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

Background

Itish was commissioned by AZNT TOKEN to perform an audit of smart contracts:
https://tronscan.org/#/contract/TD74cpU4JfMKhuWVs6uKYXKUy8nvC6EQ95/code

The purpose of the audit was to achieve the following:

- **Ensure that the smart contract functions as intended.**
- **Identify potential security issues with the smart contract.**

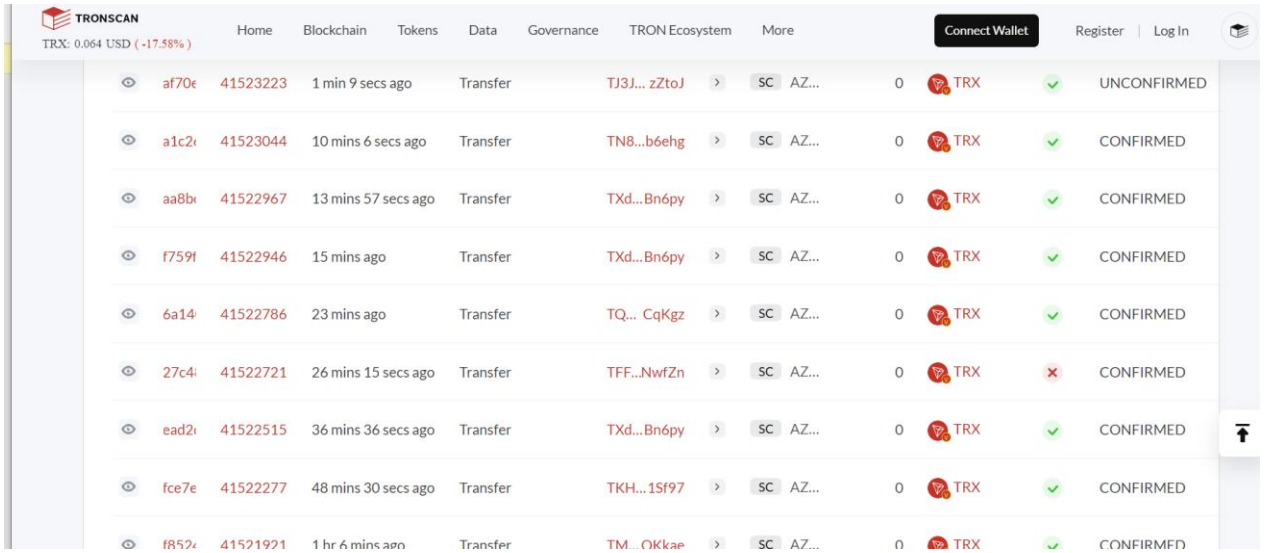
The information in this report should be used to understand the risk exposure of the smart contract, and as a guide to improve the security posture of the smart contract by remediating the issues that were identified.

Contract Details

Token contract details for 13.06.2022

Contract name	**TOKEN**
Contract address	TD74cpU4JfMKhuWVs6uKYXKUy8nvC6EQ95
Transaction's count	**282,929**
Token Creator	TFLak9BQTREUHsAQo8V5wXkcoNxev1rc3L

TOKEN Contract Top Transactions



TRONSCAN
TRX: 0.064 USD (-17.58%)
Home | Blockchain | Tokens | Data | Governance | TRON Ecosystem | More | Connect Wallet | Register | Log In

af70e	41523223	1 min 9 secs ago	Transfer	TJ3J... zZtoJ	SC AZ...	0	TRX	UNCONFIRMED
a1c2e	41523044	10 mins 6 secs ago	Transfer	TN8...b6ehg	SC AZ...	0	TRX	CONFIRMED
aa8be	41522967	13 mins 57 secs ago	Transfer	TXd...Bn6py	SC AZ...	0	TRX	CONFIRMED
f759f	41522946	15 mins ago	Transfer	TXd...Bn6py	SC AZ...	0	TRX	CONFIRMED
6a14e	41522786	23 mins ago	Transfer	TQ... CqKgz	SC AZ...	0	TRX	CONFIRMED
27c4e	41522721	26 mins 15 secs ago	Transfer	TFF...NwfZn	SC AZ...	0	TRX	CONFIRMED
ead2e	41522515	36 mins 36 secs ago	Transfer	TXd...Bn6py	SC AZ...	0	TRX	CONFIRMED
fce7e	41522277	48 mins 30 secs ago	Transfer	TKH...1Sf97	SC AZ...	0	TRX	CONFIRMED
f852e	41521921	1 hr 6 mins ago	Transfer	TM...QKkae	SC AZ...	0	TRX	CONFIRMED

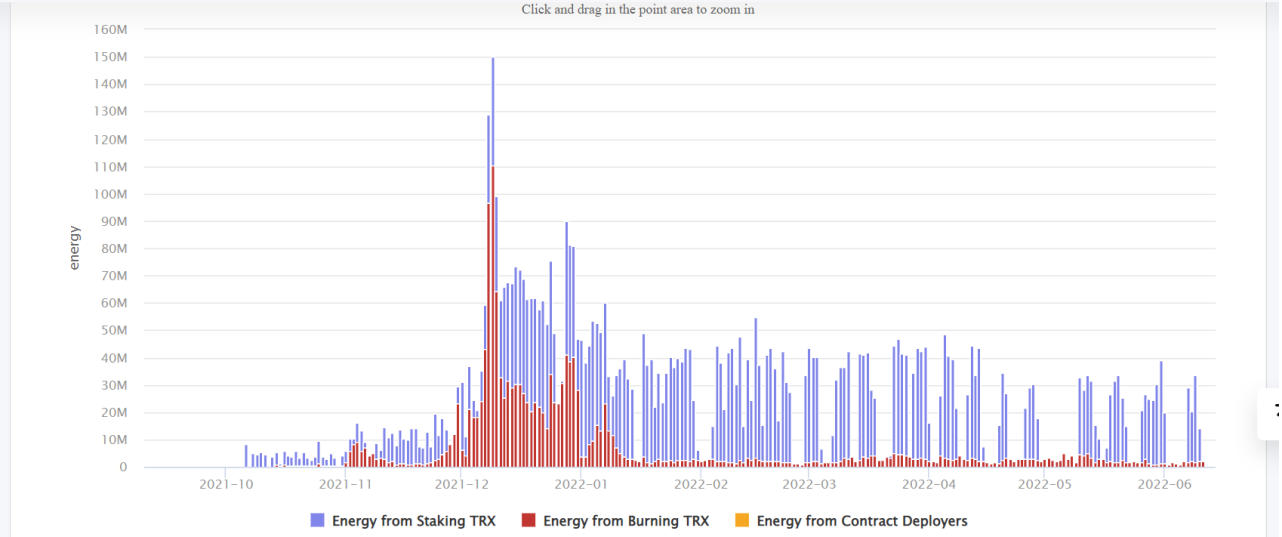
Contract Functions Details

```
totalSupply()
balanceOf
transfer
allowance
approve
transferFrom
increaseAllowance()
decreaseAllowance()
_transfer
_mint
_burn
_approve
_burnFrom
```

Issues Checking Status

	Issue description	Checking status
1.	Compiler errors.	Pass
2.	Compiler Compatibilities	Pass
3.	Possible delays in data delivery.	Pass
4.	Oracle calls.	Moderate
5.	Front running.	Pass
6.	Timestamp dependence.	Pass
7.	Integer Overflow and Underflow.	Pass
8.	DoS with Revert.	Pass
9.	DoS with block gas limit.	Moderate
10.	Methods execution permissions.	Pass
11.	Economy model of the contract.	Pass
12.	The impact of the exchange rate on the logic.	Moderate
13.	Private user data leaks.	Pass
14.	Malicious Event log.	Pass
15.	Scoping and Declarations.	Pass
16.	Uninitialized storage pointers.	Pass
17.	Arithmetic accuracy.	Pass
18.	Design Logic.	Poor
19.	Cross-function race conditions.	Pass
20.	Safe Open Zeppelin contracts implementation and usage.	Pass
21.	Fallback function security.	Failed

Energy Consumption



Code details

Erc20.sol

```
pragma solidity ^0.5.0;

import "./IERC20.sol";
import "./SafeMath.sol";

contract ERC20 is IERC20 {
    using SafeMath for uint256;

    mapping (address => uint256) private _balances;

    mapping (address => mapping (address => uint256)) private _allowances;

    uint256 private _totalSupply;

    function totalSupply() public view returns (uint256) {
        return _totalSupply;
    }

    function balanceOf(address account) public view returns (uint256) {
        return _balances[account];
    }
```

```
    function transfer(address recipient, uint256 amount) public returns (bool) {
        _transfer(msg.sender, recipient, amount);
        return true;
    }

    function allowance(address owner, address spender) public view returns
(uint256) {
        return _allowances[owner][spender];
    }

    function approve(address spender, uint256 value) public returns (bool) {
        _approve(msg.sender, spender, value);
        return true;
    }

    function transferFrom(address sender, address recipient, uint256 amount) public
returns (bool) {
        _transfer(sender, recipient, amount);
        _approve(sender, msg.sender, _allowances[sender][msg.sender].sub(amount));
        return true;
    }

    function increaseAllowance(address spender, uint256 addedValue) public returns
(bool) {
        _approve(msg.sender, spender,
_allowances[msg.sender][spender].add(addedValue));
        return true;
    }

    function decreaseAllowance(address spender, uint256 subtractedValue) public
returns (bool) {
        _approve(msg.sender, spender,
_allowances[msg.sender][spender].sub(subtractedValue));
        return true;
    }

    function _transfer(address sender, address recipient, uint256 amount) internal
{
        require(sender != address(0), "ERC20: transfer from the zero address");
        require(recipient != address(0), "ERC20: transfer to the zero address");

        _balances[sender] = _balances[sender].sub(amount);
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }

    function _mint(address account, uint256 amount) internal {
        require(account != address(0), "ERC20: mint to the zero address");
```

```
        _totalSupply = _totalSupply.add(amount);
        _balances[account] = _balances[account].add(amount);
        emit Transfer(address(0), account, amount);
    }

    function _burn(address account, uint256 value) internal {
        require(account != address(0), "ERC20: burn from the zero address");

        _totalSupply = _totalSupply.sub(value);
        _balances[account] = _balances[account].sub(value);
        emit Transfer(account, address(0), value);
    }

    function _approve(address owner, address spender, uint256 value) internal {
        require(owner != address(0), "ERC20: approve from the zero address");
        require(spender != address(0), "ERC20: approve to the zero address");

        _allowances[owner][spender] = value;
        emit Approval(owner, spender, value);
    }

    function _burnFrom(address account, uint256 amount) internal {
        _burn(account, amount);
        _approve(account, msg.sender,
_allowances[account][msg.sender].sub(amount));
    }
}
```

Erc20D.sol

```
pragma solidity ^0.5.0;

import "./IERC20.sol";

contract ERC20Detailed is IERC20 {
    string private _name;
    string private _symbol;
    uint8 private _decimals;

    constructor (string memory name, string memory symbol, uint8 decimals) public {
        _name = name;
        _symbol = symbol;
        _decimals = decimals;
    }

    function name() public view returns (string memory) {
        return _name;
    }
```

```
    function symbol() public view returns (string memory) {
        return _symbol;
    }

    function decimals() public view returns (uint8) {
        return _decimals;
    }
}
```

Erc20.sol

```
pragma solidity ^0.5.0;

interface IERC20 {
    function totalSupply() external view returns (uint256);

    function balanceOf(address account) external view returns (uint256);

    function transfer(address recipient, uint256 amount) external returns (bool);

    function allowance(address owner, address spender) external view returns
(uint256);

    function approve(address spender, uint256 amount) external returns (bool);

    function transferFrom(address sender, address recipient, uint256 amount)
external returns (bool);

    event Transfer(address indexed from, address indexed to, uint256 value);

    event Approval(address indexed owner, address indexed spender, uint256 value);
}
```

SafeMath

```
pragma solidity ^0.5.0;

library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");

        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b <= a, "SafeMath: subtraction overflow");
        uint256 c = a - b;
```

```
        return c;
    }

    function mul(uint256 a, uint256 b) internal pure returns (uint256) {
        if (a == 0) {
            return 0;
        }

        uint256 c = a * b;
        require(c / a == b, "SafeMath: multiplication overflow");

        return c;
    }

    function div(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b > 0, "SafeMath: division by zero");
        uint256 c = a / b;

        return c;
    }

    function mod(uint256 a, uint256 b) internal pure returns (uint256) {
        require(b != 0, "SafeMath: modulo by zero");
        return a % b;
    }
}
```

Token.sol

```
pragma solidity ^0.5.0;

import "./ERC20.sol";
import "./ERC20Detailed.sol";

contract Token is ERC20, ERC20Detailed {
    constructor () public ERC20Detailed("AZNT", "AZNT", 6) {
        _mint(msg.sender, 2000000000 * (10 ** uint256(decimals())));
    }
}
```

Security Issues

High Severity Issues

No high severity issues found.

Medium Severity Issues

No medium severity issues found.

Low Severity Issues

Low severity issues found in logic of codes.

Token.sol

```
pragma solidity ^0.5.0;

import "./ERC20.sol";
import "./ERC20Detailed.sol";

contract Token is ERC20, ERC20Detailed {
    constructor () public ERC20Detailed("AZNT", "AZNT", 6) {
        _mint(msg.sender, 2000000000 * (10 ** uint256(decimals())));
    }
}
```

Owner Owned This Much Supply Of Tokens While Deploying This Contract ……

& Token Locking Details Are Not Provided By The Team Which Could Result In A Large Dumping Of The Token

Conclusion

Smart contracts contain High severity issues! Liquidity pair contract's security is not checked due to out of scope.

Liquidity locking details NOT provided by the team.

Itish note:

Please check the disclaimer above and note, the audit makes no statements or warranties on business model, investment attractiveness or code sustainability. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed