Audit Report

TheX protocol

November 2023

Network	BSC
Address	0x91829779e4a3c23f96f006e5e47a995985adab05
Audited by	© cyberscope

Analysis

● Critical ● Medium ● Minor / Informative ● Pass

Severity	Code	Description	Status
●	ST	Stops Transactions	Unresolved
●	OTUT	Transfers User's Tokens	Passed
●	ELFM	Exceeds Fees Limit	Passed
●	MT	Mints Tokens	Unresolved
●	BT	Burns Tokens	Unresolved
●	BC	Blacklists Addresses	Unresolved

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	RM	Redundant Modifier	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L06	Missing Events Access Control	Unresolved
●	L11	Unnecessary Boolean equality	Unresolved
●	L15	Local Scope Variable Shadowing	Unresolved
●	L16	Validate Variable Setters	Unresolved
●	L19	Stable Compiler Version	Unresolved
●	L22	Potential Locked Ether	Unresolved

Table of Contents

Review

Contract Name	TheXProtocol
Compiler Version	v0.8.17+commit.8df45f5f
Optimization	200 runs
Explorer	https://bscscan.com/address/0x91829779e4a3c23f96f006e5e47a995985adab05
Address	0x91829779e4a3c23f96f006e5e47a995985adab05
Network	BSC
Symbol	TXP
Decimals	18
Total Supply	166,113,475

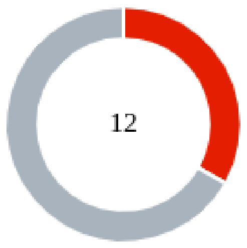
Audit Updates

Initial Audit	22 Nov 2023

Source Files

Filename	SHA256
TheXProtocol.sol	35035a2882b3f43b1f420339c8807b2d43f8e8122abf3546ee04eb564b4aa6d1

Findings Breakdown



●	Critical	4
●	Medium	0
●	Minor / Informative	8

Severity	Unresolved	Acknowledged	Resolved	Other
● Critical	4	0	0	0
● Medium	0	0	0	0
● Minor / Informative	8	0	0	0

ST - Stops Transactions

Criticality	Critical
Location	TheXProtocol.sol#L155,208,479,764
Status	Unresolved

Description

The transactions are initially disabled for all users excluding the authorized addresses. The owner can enable the transactions for all users by invoking the `_finishPresale` function. Once the transactions are enable the owner will not be able to disable them again.

```
   bool isPresaleDone = false;

     modifier isPresaleLive {
        if (isPresaleDone == true || _msgSender() ==
_distributionOwner || txpwallets[msg.sender] ) {
        _;
        } else {
        revert('The Pre-Sale is Still Ongoing');
        }
    }

    function transfer(
        address to,
        uint256 amount
    ) public virtual isPresaleLive returns (bool)  {
        address owner = _msgSender();
        _transfer(owner, to, amount);
        return true;
    }

    function _finishPresale () external virtual onlyOwner {
        isPresaleDone = true;
        emit PresaleDone();
    }
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions. Some suggestions are:

- Introduce a multi-sign wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

MT - Mints Tokens

Criticality	Critical
Location	TheXProtocol.sol#L640
Status	Unresolved

Description

The Mint Owner role has the authority to mint tokens. The minter account may take advantage of it by calling the `_mint` function. As a result, the contract tokens will be highly inflated.

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

        _beforeTokenTransfer(address(0), account, amount);

        _totalSupply += amount;
        _balances[account] += amount;

        emit Mint(account, amount);
        emit Transfer(address(0), account, amount);
    }
```

Recommendation

The team should carefully manage the private keys of the mint's owner account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.

- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BT - Burns Tokens

Criticality	Critical
Location	TheXProtocol.sol#L663
Status	Unresolved

Description

The Burn Owner role has the authority to burn tokens from a specific address. The burner account may take advantage of it by calling the `_burn` function. As a result, the targeted address will lose the corresponding tokens.

```
    function _burn(address account, uint256 amount) public
virtual onlyBurnOwner {
        require(account != address(0), "ERC20: burn from the
zero address");

        _beforeTokenTransfer(account, address(0), amount);

        uint256 accountBalance = _balances[account];
        require(accountBalance >= amount, "ERC20: burn amount
exceeds balance");

        _balances[account] = accountBalance - amount;
        _totalSupply -= amount;

        emit Transfer(account, address(0), amount);

        //  _afterTokenTransfer(account, address(0), amount);
    }
```

Recommendation

The team should carefully manage the private keys of the owner’s account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

BC - Blacklists Addresses

Criticality	Critical
Location	TheXProtocol.sol#L334
Status	Unresolved

Description

The contract owner has the authority to stop addresses from transactions. The owner may take advantage of it by calling the `addAddressToBlacklist` function.

```
    function addAddressToBlacklist(address _user) public
onlyOwner {
        require(!isBlacklisted[_user], "User Already
Blacklisted");
        isBlacklisted[_user] = true;
    }
```

Recommendation

The team should carefully manage the private keys of the owner's account. We strongly recommend a powerful security mechanism that will prevent a single user from accessing the contract admin functions.

Temporary Solutions:

These measurements do not decrease the severity of the finding

- Introduce a time-locker mechanism with a reasonable delay.
- Introduce a multi-signature wallet so that many addresses will confirm the action.
- Introduce a governance model where users will vote about the actions.

Permanent Solution:

- Renouncing the ownership, which will eliminate the threats but it is non-reversible.

RM - Redundant Modifier

Criticality	Minor / Informative
Location	TheXProtocol.sol#L201
Status	Unresolved

Description

There are code segments that could be optimized. A segment may be optimized so that it becomes a smaller size, consumes less memory, executes more rapidly, or performs fewer operations.

The contract declares the `onlyDistribution` modifier. The modifier is not being used by the contract. As a result, the modifier is redundant.

```
    modifier onlyDistribution  {
        require(_distributionOwner != address(0),'Distribution
address is not set!');
        require(msg.sender == _distributionOwner,'Not
authorized!');
        _;
    }
```

Recommendation

The team is advised to take these segments into consideration and rewrite them so the runtime will be more performant. That way it will improve the efficiency and performance of the source code and reduce the cost of executing it. If the intended purpose of the code is to utilize the `tokensTransferable` modifier, then the contract should incorporate it. Otherwise, since the modifier is not used within the contract, it can be safely removed to streamline the code.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	TheXProtocol.sol#L170,243,334,339,344,640,663,764
Status	Unresolved

Description

The Solidity style guide is a set of guidelines for writing clean and consistent Solidity code. Adhering to a style guide can help improve the readability and maintainability of the Solidity code, making it easier for others to understand and work with.

The followings are a few key points from the Solidity style guide:

1. Use camelCase for function and variable names, with the first letter in lowercase (e.g., myVariable, updateCounter).
2. Use PascalCase for contract, struct, and enum names, with the first letter in uppercase (e.g., MyContract, UserStruct, ErrorEnum).
3. Use uppercase for constant variables and enums (e.g., MAX_VALUE, ERROR_CODE).
4. Use indentation to improve readability and structure.
5. Use spaces between operators and after commas.
6. Use comments to explain the purpose and behavior of the code.
7. Keep lines short (around 120 characters) to improve readability.

```
address _address
address _user

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

        _beforeTokenTransfer(address(0), account, amount);

        _totalSupply += amount;
        _balances[account] += amount;

        emit Mint(account, amount);
        emit Transfer(address(0), account, amount);
    }

...
```

Recommendation

By following the Solidity naming convention guidelines, the codebase increased the readability, maintainability, and makes it easier to work with.
Find more information on the Solidity documentation https://docs.soliditylang.org/en/v0.8.17/style-guide.html#naming-convention.

L06 - Missing Events Access Control

Criticality	Minor / Informative
Location	TheXProtocol.sol#L296,305,314
Status	Unresolved

Description

Events are a way to record and log information about changes or actions that occur within a contract. They are often used to notify external parties or clients about events that have occurred within the contract, such as the transfer of tokens or the completion of a task. There are functions that have no event emitted, so it is difficult to track off-chain changes.

```
_mintOwner = mintOwner
_burnOwner = burnOwner
_distributionOwner = distributionOwner
```

Recommendation

To avoid this issue, it's important to carefully design and implement the events in a contract, and to ensure that all required events are included. It's also a good idea to test the contract to ensure that all events are being properly triggered and logged.

By including all required events in the contract and thoroughly testing the contract's functionality, the contract ensures that it performs as intended and does not have any missing events that could cause issues.

L11 - Unnecessary Boolean equality

Criticality	Minor / Informative
Location	TheXProtocol.sol#L209
Status	Unresolved

Description

Boolean equality is unnecessary when comparing two boolean values. This is because a boolean value is either true or false, and there is no need to compare two values that are already known to be either true or false.

it's important to be aware of the types of variables and expressions that are being used in the contract's code, as this can affect the contract's behavior and performance. The comparison to boolean constants is redundant. Boolean constants can be used directly and do not need to be compared to true or false.

```
isPresaleDone == true || _msgSender() == _distributionOwner ||
txpwallets[msg.sender]
```

Recommendation

Using the boolean value itself is clearer and more concise, and it is generally considered good practice to avoid unnecessary boolean equalities in Solidity code.

L15 - Local Scope Variable Shadowing

Criticality	Minor / Informative
Location	TheXProtocol.sol#L408,790,791
Status	Unresolved

Description

Local scope variable shadowing occurs when a local variable with the same name as a variable in an outer scope is declared within a function or code block. When this happens, the local variable "shadows" the outer variable, meaning that it takes precedence over the outer variable within the scope in which it is declared.

```
address _owner
string memory name
string memory symbol
```

Recommendation

It's important to be aware of shadowing when working with local variables, as it can lead to confusion and unintended consequences if not used correctly. It's generally a good idea to choose unique names for local variables to avoid shadowing outer variables and causing confusion.

L16 - Validate Variable Setters

Criticality	Minor / Informative
Location	TheXProtocol.sol#L296,305,314
Status	Unresolved

Description

The contract performs operations on variables that have been configured on user-supplied input. These variables are missing of proper check for the case where a value is zero. This can lead to problems when the contract is executed, as certain actions may not be properly handled when the value is zero.

```
_mintOwner = mintOwner
_burnOwner = burnOwner
_distributionOwner = distributionOwner
```

Recommendation

By adding the proper check, the contract will not allow the variables to be configured with zero value. This will ensure that the contract can handle all possible input values and avoid unexpected behavior or errors. Hence, it can help to prevent the contract from being exploited or operating unexpectedly.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	TheXProtocol.sol#L2,87,113,352
Status	Unresolved

Description

The `^` symbol indicates that any version of Solidity that is compatible with the specified version (i.e., any version that is a higher minor or patch version) can be used to compile the contract. The version lock is a mechanism that allows the author to specify a minimum version of the Solidity compiler that must be used to compile the contract code. This is useful because it ensures that the contract will be compiled using a version of the compiler that is known to be compatible with the code.

```
pragma solidity ^0.8.4;
```

Recommendation

The team is advised to lock the pragma to ensure the stability of the codebase. The locked pragma version ensures that the contract will not be deployed with an unexpected version. An unexpected version may produce vulnerabilities and undiscovered bugs. The compiler should be configured to the lowest version that provides all the required functionality for the codebase. As a result, the project will be compiled in a well-tested LTS (Long Term Support) environment.

L22 - Potential Locked Ether

Criticality	Minor / Informative
Location	TheXProtocol.sol#L789
Status	Unresolved

Description

The contract contains Ether that has been placed into a Solidity contract and is unable to be transferred. Thus, it is impossible to access the locked Ether. This may produce a financial loss for the users that have called the payable method.

```
constructor(
        string memory name,
        string memory symbol,
        uint256 initialBalance,
        address owneraddress,
        uint8 decimals_
    ) payable ERC20(name, symbol, initialBalance, owneraddress,
decimals_) {}
```

Recommendation

The team is advised to either remove the payable method or add a withdraw functionality. it is important to carefully consider the risks and potential issues associated with locked Ether.

Functions Analysis

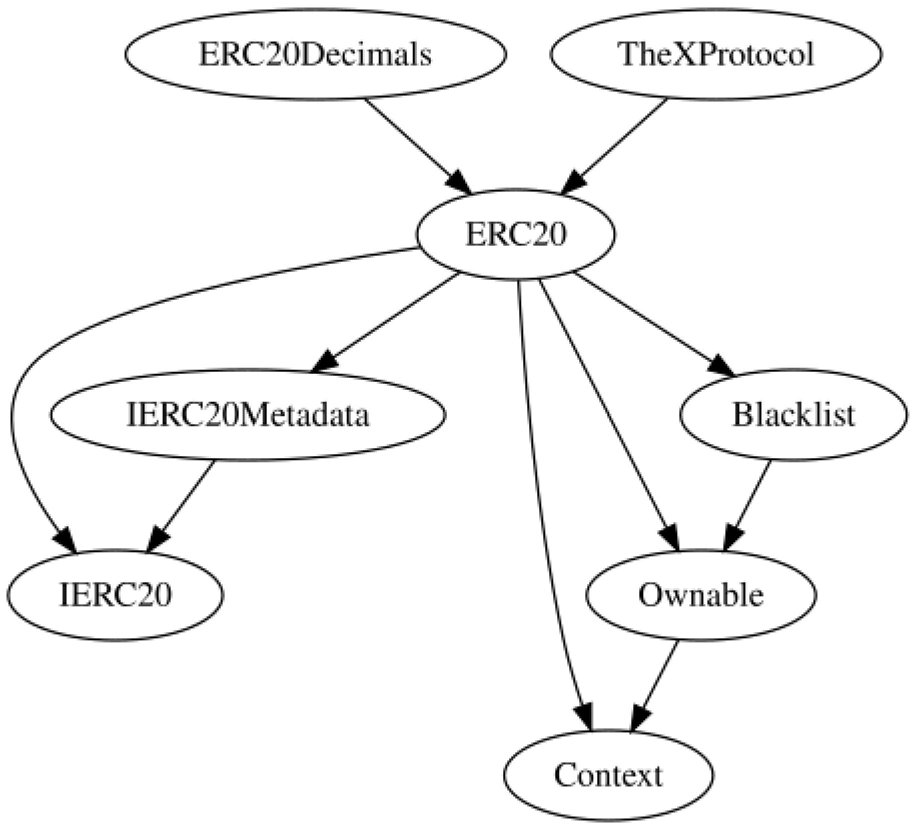
Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
IERC20	Interface			
	totalSupply	External		-
	balanceOf	External		-
	transfer	External	✓	-
	allowance	External		-
	approve	External	✓	-
	transferFrom	External	✓	-
IERC20Metadata	Interface	IERC20		
	name	External		-
	symbol	External		-
	decimals	External		-
Context	Implementation			
	_msgSender	Internal		
	_msgData	Internal		
Ownable	Implementation	Context		

	addAddressToTokenomicsWallet	Public	✓	onlyOwner
		Public	✓	-
	owner	Public		-
	getMintOwner	Public		-
	getBurnOwner	Public		-
	getDistributionOwner	Public		-
	getTokenomicsWallets	Public		-
	_checkOwner	Internal		
	renounceOwnership	Public	✓	onlyOwner
	transferOwnership	Public	✓	onlyOwner
	_transferOwnership	Internal	✓	
	setMintOwner	Public	✓	onlyOwner
	setBurnOwner	Public	✓	onlyOwner
	setDistributionOwner	Public	✓	onlyOwner
Blacklist	Implementation	Ownable		
	addAddressToBlacklist	Public	✓	onlyOwner
	removeFromBlacklist	Public	✓	onlyOwner
	getBlacklistUser	Public		-
ERC20	Implementation	Context, IERC20, IERC20Metadata, Ownable, Blacklist		
		Public	✓	-

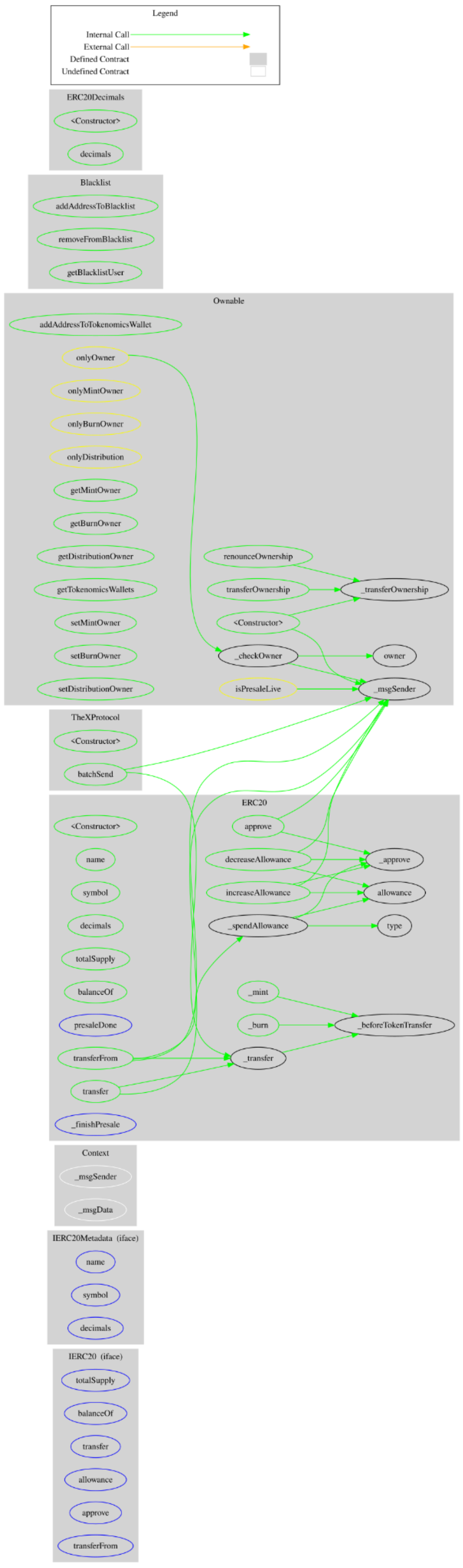
	name	Public		-
	symbol	Public		-
	decimals	Public		-
	totalSupply	Public		-
	balanceOf	Public		-
	presaleDone	External		-
	transfer	Public	✓	isPresaleLive
	allowance	Public		-
	approve	Public	✓	-
	transferFrom	Public	✓	-
	increaseAllowance	Public	✓	-
	decreaseAllowance	Public	✓	-
	_transfer	Internal	✓	
	_mint	Public	✓	onlyMintOwner
	_burn	Public	✓	onlyBurnOwner
	_approve	Internal	✓	
	_spendAllowance	Internal	✓	
	_beforeTokenTransfer	Internal	✓	
	_finishPresale	External	✓	onlyOwner
ERC20Decimals	Implementation	ERC20		
		Public	✓	-
	decimals	Public		-

TheXProtocol	Implementation	ERC20		
		Public	Payable	ERC20
	batchSend	Public	✓	onlyOwner

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
ERC20Decimals
<Constructor>
decimals
Blacklist
addAddressToBlacklist
removeFromBlacklist
getBlacklistUser
Ownable
addAddressToTokenomicsWallet
onlyOwner
onlyMintOwner
onlyBurnOwner
onlyDistribution
getMintOwner
getBurnOwner
getDistributionOwner
getTokenomicsWallets
setMintOwner
setBurnOwner
setDistributionOwner
renounceOwnership
transferOwnership
<Constructor>
_transferOwnership
_checkOwner
owner
isPresaleLive
_msgSender
TheXProtocol
<Constructor>
batchSend
ERC20
<Constructor>
name
symbol
decimals
totalSupply
balanceOf
presaleDone
transferFrom
transfer
_finishPresale
approve
decreaseAllowance
increaseAllowance
_spendAllowance
_approve
allowance
type
_mint
_burn
_beforeTokenTransfer
_transfer
Context
_msgSender
_msgData
IERC20Metadata (iface)
name
symbol
decimals
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom

Summary

TheX protocol contract implements a token mechanism. This audit investigates security issues, business logic concerns and potential improvements. There are some functions that can be abused by the owner like stop transactions, mint tokens, burn tokens from any address and massively blacklist addresses. If the contract owner abuses the mint functionality, then the contract will be highly inflated. If the contract owner abuses the burn functionality, then the users could lose their tokens. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

Disclaimer

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io