Cyberscope

A TAC Security Company

Audit Report

XITCOIN

July 2025

SHA256

64f103dd1d9b6d6b4c72c22a4c3dbe8982723a7ae201a8158e099ca224ed26c6

Audited by © cyberscope

Analysis

🔴 Critical 🟠 Medium ⚪ Minor / Informative 🔵 Pass

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

Diagnostics

● Critical ● Medium ● Minor / Informative

Severity	Code	Description	Status
●	PGM	Potential Griefing Manipulation	Unresolved
●	MEE	Missing Events Emission	Unresolved
●	MU	Modifiers Usage	Unresolved
●	RRO	Revocable Renounce Operation	Unresolved
●	UUI	Unsanitized User Input	Unresolved
●	L04	Conformance to Solidity Naming Conventions	Unresolved
●	L19	Stable Compiler Version	Unresolved

Table of Contents

Analysis **1**
Diagnostics **2**
Table of Contents **3**
Risk Classification **4**
Review **5**
Audit Updates 5
Source Files 5
Findings Breakdown **6**
PGM - Potential Griefing Manipulation 7
Description 7
Recommendation 8
MEE - Missing Events Emission 9
Description 9
Recommendation 9
MU - Modifiers Usage 10
Description 10
Recommendation 10
RRO - Revocable Renounce Operation 11
Description 11
Recommendation 11
UUI - Unsanitized User Input 12
Description 12
Recommendation 12
L04 - Conformance to Solidity Naming Conventions 13
Description 13
Recommendation 13
L19 - Stable Compiler Version 14
Description 14
Recommendation 14
Functions Analysis **15**
Inheritance Graph **16**
Flow Graph **17**
Summary **18**
Disclaimer **19**
About Cyberscope **20**

Risk Classification

The criticality of findings in Cyberscope's smart contract audits is determined by evaluating multiple variables. The two primary variables are:

1. **Likelihood of Exploitation**: This considers how easily an attack can be executed, including the economic feasibility for an attacker.
2. **Impact of Exploitation**: This assesses the potential consequences of an attack, particularly in terms of the loss of funds or disruption to the contract's functionality.

Based on these variables, findings are categorized into the following severity levels:

1. **Critical**: Indicates a vulnerability that is both highly likely to be exploited and can result in significant fund loss or severe disruption. Immediate action is required to address these issues.
2. **Medium**: Refers to vulnerabilities that are either less likely to be exploited or would have a moderate impact if exploited. These issues should be addressed in due course to ensure overall contract security.
3. **Minor**: Involves vulnerabilities that are unlikely to be exploited and would have a minor impact. These findings should still be considered for resolution to maintain best practices in security.
4. **Informative**: Points out potential improvements or informational notes that do not pose an immediate risk. Addressing these can enhance the overall quality and robustness of the contract.

Severity	Likelihood / Impact of Exploitation
● Critical	Highly Likely / High Impact
● Medium	Less Likely / High Impact or Highly Likely/ Lower Impact
● Minor / Informative	Unlikely / Low to no Impact

Review

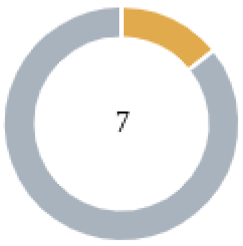
Audit Updates

Initial Audit	02 Jul 2025

Source Files

Filename	SHA256
latestXTCUP.sol	64f103dd1d9b6d6b4c72c22a4c3dbe8982723a7ae201a8158e099ca224ed26c6

Findings Breakdown



●	Critical	0
●	Medium	1
●	Minor / Informative	6

Severity		Unresolved	Acknowledged	Resolved	Other
●	Critical	0	0	0	0
●	Medium	1	0	0	0
●	Minor / Informative	6	0	0	0

PGM - Potential Griefing Manipulation

Criticality	Medium
Location	latestXTCUP.sol#L45,74
Status	Unresolved

Description

The contract implements a voting mechanism to manage upgrade and ownership update operations, requiring 2 out of 3 votes for approval. However, this mechanism is vulnerable to griefing attacks by any of the voters. Specifically, a malicious voter can submit a proposal that differs from the current `pendingUpgrade` or `pendingOwner` , which resets the entire voting process. As a result, a rogue voter can continuously disrupt the mechanism, effectively bypassing the intended 2-out-of-3 consensus design.

Such griefing attacks could undermine the contract's usability and obstruct legitimate operations.

```
function voteToUpgrade(address newImplementation) public onlyVoter {
require(newImplementation != address(0), "Invalid new
implementation");
if (pendingUpgrade != newImplementation) {
_resetUpgradeVotes();
pendingUpgrade = newImplementation;
}
...
}
```

```
function voteToChangeOwner(address newOwner) public onlyVoter {
require(newOwner != address(0), "Invalid new owner");
if (pendingOwner != newOwner) {
_resetOwnerChangeVotes();
pendingOwner = newOwner;
}
...
}
```

Recommendation

The team is advised to review the voting mechanism to ensure that all legitimate operations are processed as intended. The contract should handle updates of the current voting process gracefully. The team could consider timelocking or majority mechanisms. This will help maintain the integrity of user activities and strengthen trust in the system.

MEE - Missing Events Emission

Criticality	Minor / Informative
Location	latestXTCUP.sol#L23,45,74
Status	Unresolved

Description

The contract performs actions and state mutations from external methods that do not result in the emission of events. Emitting events for significant actions is important as it allows external parties, such as wallets or dApps, to track and monitor the activity on the contract. Without these events, it may be difficult for external parties to accurately determine the current state of the contract.

```
voters = votersAdmin;
```

Recommendation

It is recommended to include events in the code that are triggered each time a significant action is taking place within the contract. These events should include relevant details such as the user's address and the nature of the action taken. By doing so, the contract will be more transparent and easily auditable by external parties. It will also help prevent potential issues or disputes that may arise in the future.

MU - Modifiers Usage

Criticality	Minor / Informative
Location	latestXTCUP.sol#L46,75
Status	Unresolved

Description

The contract is using repetitive statements on some methods to validate some preconditions. In Solidity, the form of preconditions is usually represented by the modifiers. Modifiers allow you to define a piece of code that can be reused across multiple functions within a contract. This can be particularly useful when you have several functions that require the same checks to be performed before executing the logic within the function.

```
require(newImplementation != address(0), "Invalid new
implementation");
require(newOwner != address(0), "Invalid new owner");
```

Recommendation

The team is advised to use modifiers since it is a useful tool for reducing code duplication and improving the readability of smart contracts. By using modifiers to perform these checks, it reduces the amount of code that is needed to write, which can make the smart contract more efficient and easier to maintain.

RRO - Revocable Renounce Operation

Criticality	Minor / Informative
Location	latestXTCUP.sol#L74
Status	Unresolved

Description

The contract inherits the `OwnableUpgradeable` extension and implements the `renounceOwnership` method. However, the implementation of `voteToChangeOwner` overrides the effects of a potential renouncement by allowing ownership to be reassigned from the zero address to a specified address. This behavior may contradict the intended purpose of the `renounceOwnership` function, which is used to permanently revoke ownership rights.

```
function voteToChangeOwner(address newOwner) public onlyVoter {
...
}
```

Recommendation

The team is advised to enhance the consistency of the contract's ownership logic. If the intended design is to prevent renouncement, the `renounceOwnership` method could be overridden. Alternatively, the contract should enforce restrictions to prevent ownership transfer once it has been delegated to the zero address. This will improve consistency and increase user trust in the system.

UUI - Unsanitized User Input

Criticality	Minor / Informative
Location	latestXTCUP.sol#L23
Status	Unresolved

Description

The contract accepts user information without validating that it conforms to the proper structure. Specifically, the contract does not ensure variables that are provided by the user to not retain zero values.

```
voters = votersAdmin;
```

Recommendation

The team is advised to properly check the variables according to the required specifications. This will ensure the necessary information is provided enhancing the contract's consistency.

L04 - Conformance to Solidity Naming Conventions

Criticality	Minor / Informative
Location	latestXTCUP.sol#L18,21,22,28,44,51,55,57,61,65,68
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.

```
mapping(address => bool) public ownerChangeVotes;
uint8 public ownerChangeVoteCount;
address public pendingOwner;
```

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/stable/style-guide.html#naming-conventions.

L19 - Stable Compiler Version

Criticality	Minor / Informative
Location	latestXTCUP.sol#L4
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 ^0.8.30;
```

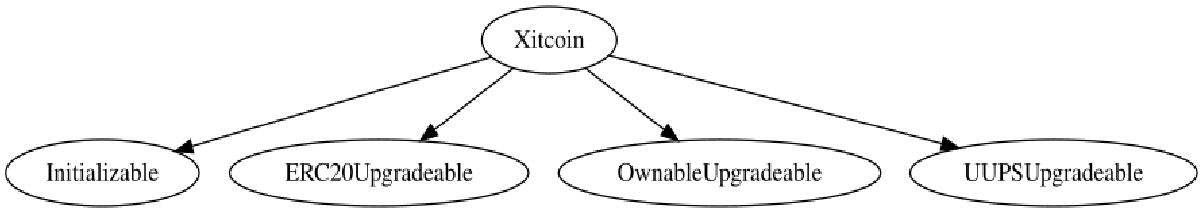
Recommendation

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.

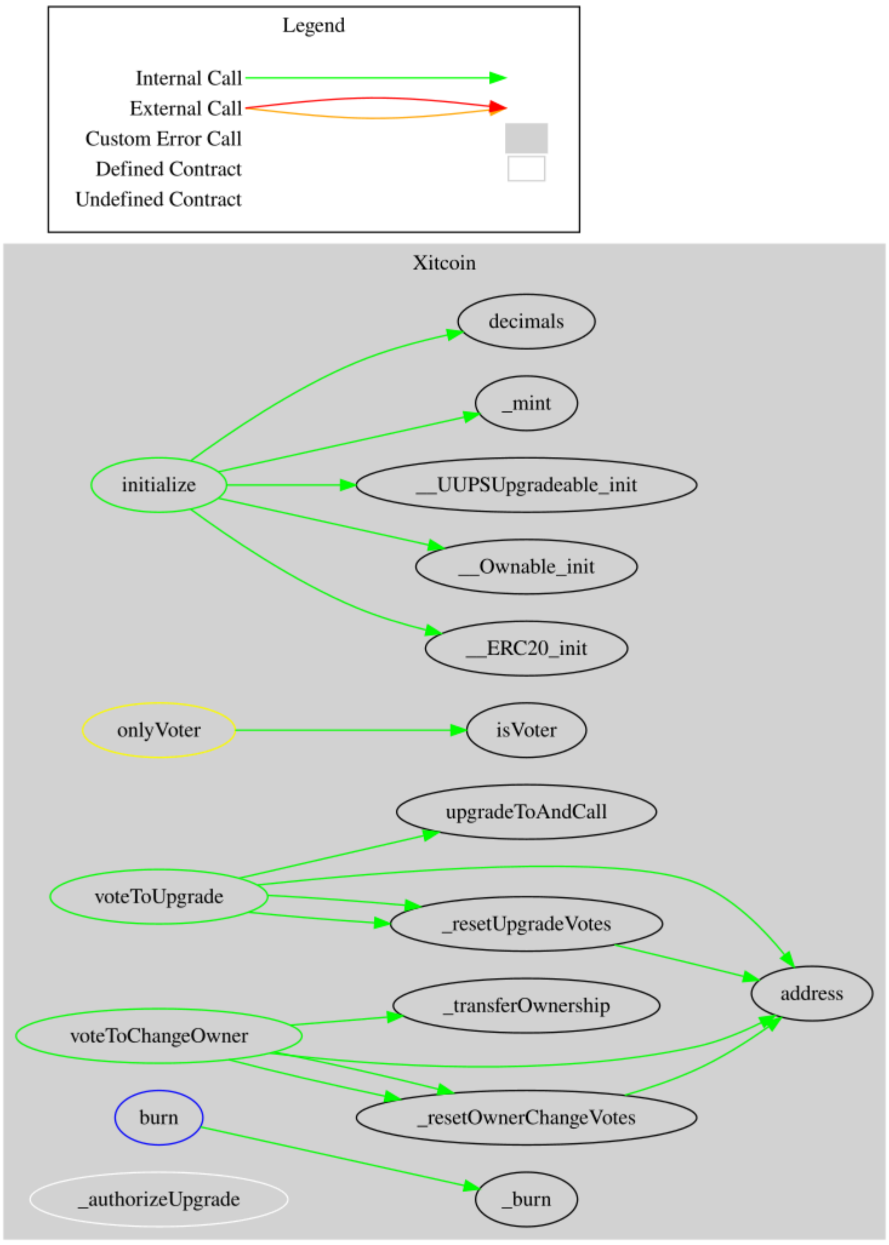
Functions Analysis

Contract	Type	Bases		
	Function Name	**Visibility**	**Mutability**	**Modifiers**
Xitcoin	Implementation	Initializable, ERC20Upgradeable, OwnableUpgradeable, UUPSUpgradeable		
	initialize	Public	✓	initializer
	isVoter	Public		-
	voteToUpgrade	Public	✓	onlyVoter
	_resetUpgradeVotes	Internal	✓	
	voteToChangeOwner	Public	✓	onlyVoter
	_resetOwnerChangeVotes	Internal	✓	
	burn	External	✓	onlyOwner
	_authorizeUpgrade	Internal		

Inheritance Graph



Flow Graph


Legend
Internal Call
External Call
Custom Error Call
Defined Contract
Undefined Contract
Xitcoin
decimals
_mint
initialize
__UUPSUpgradeable_init
__Ownable_init
__ERC20_init
onlyVoter
isVoter
upgradeToAndCall
voteToUpgrade
_resetUpgradeVotes
address
_transferOwnership
voteToChangeOwner
burn
_resetOwnerChangeVotes
_authorizeUpgrade
_burn

Summary

XITCOIN contract implements a token and governance mechanism. This audit investigates security issues, business logic concerns and potential improvements. XITCOIN is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an “endorsement” or “disapproval” of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a TAC 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.

A *TAC Security* Company

The Cyberscope team

cyberscope.io