Approved

SMART CONTRACT SECURITY AUDIT

CYRUS

Scan and check this report was posted at Approved Github

December, 2023

Website: Approved.ltd

Table of Contents

Disclaimer

This is a comprehensive report based on our automated and manual examination of cybersecurity vulnerabilities and framework flaws of the project's smart contract.

Reading the full analysis report is essential to build your understanding of project's security level. It is crucial to take note, though we have done our best to perform this analysis and report, that you should not rely on the our research and cannot claim what it states or how we created it.

Before making any judgments, you have to conduct your own independent research.

We will discuss this in more depth in the following disclaimer - please read it fully.

DISCLAIMER: You agree to the terms of this disclaimer by reading this report or any portion thereof. Please stop reading this report and remove and delete any copies of this report that you download and/or print if you do not agree to these conditions. Scan and verify report's presence in the GitHub repository by a qr-code at the title page. This report is for non-reliability information only and does not represent investment advice. No one shall be entitled to depend on the report or its contents, and Approved and its affiliates shall not be held responsible to you or anyone else, nor shall Approved provide any guarantee or representation to any person with regard to the accuracy or integrity of the report.

Without any terms, warranties or other conditions other than as set forth in that exclusion and Approved excludes hereby all representations, warrants, conditions and other terms (including, without limitation, guarantees implied by the law of satisfactory quality, fitness for purposes and the use of reasonable care and skills).

The report is provided as "as is" and does not contain any terms and conditions. Except as legally banned, Approved disclaims all responsibility and responsibilities and no claim against Approved is made to any amount or type of loss or damages (without limitation, direct, indirect, special, punitive, consequential or pure economic loses or losses) that may be caused by you or any other person, or any damages or damages, including without limitations (whether innocent or negligent).

Security analysis is based only on the smart contracts. No applications or operations were reviewed for security. No product code has been reviewed.

Procedure

Our analysis contains following steps:

1. Project Analysis;

2. Unit Testing:
   - Smart contract functions will be unit tested on multiple parameters and under multiple conditions to ensure that all paths of functions are functioning as intended.
   - In this phase intended behaviour of smart contract is verified.
   - In this phase, we would also ensure that smart contract functions are not consuming unnecessary gas.
   - Gas limits of functions will be verified in this stage.

3. Automated Testing:
   - Mythril
   - Oyente
   - Manticore
   - Solgraph

Terminology

We categorize the finding into 4 categories based on their vulnerability:

- Low-severity issue — less important, must be analyzed
- Medium-severity issue — important, needs to be analyzed and fixed
- High-severity issue —important, might cause vulnerabilities, must be analyzed and fixed
- Critical-severity issue —serious bug causes, must be analyzed and fixed.

Limitations

The security audit of Smart Contract cannot cover all vulnerabilities. Even if no vulnerabilities are detected in the audit, there is no guarantee that future smart contracts are safe. Smart contracts are in most cases safeguarded against specific sorts of attacks. In order to find as many flaws as possible, we carried out a comprehensive smart contract audit. Audit is a document that is not legally binding and guarantees nothing.

Basic Security Recommendation

Unlike hardware and paper wallets, hot wallets are connected to the internet and store private keys online, which exposes them to greater risk. If a company or an individual holds significant amounts of cryptocurrency in a hot wallet, they should consider using MultiSig addresses. Wallet security is enhanced when private keys are stored in different locations and are not controlled by a single entity.

Token Contract Details for 07.12.2023

Deployed address: **0xb853f7852Aa780831F165899ccbAF5DB0882B0d6**

Total Supply: **3,000,000,000**

Token Tracker: **CRS**

Token holders: **3**

Transactions count: **5**

Top 100 holders dominance: **100.00%**

Audit Details

Project Name: **CYRUS**

Language: **Solidity**

Compiler Version: **v0.8.22**

Blockchain: **Polygonscan**

Social Profiles

Project Website: **https://cyruslab.io/**

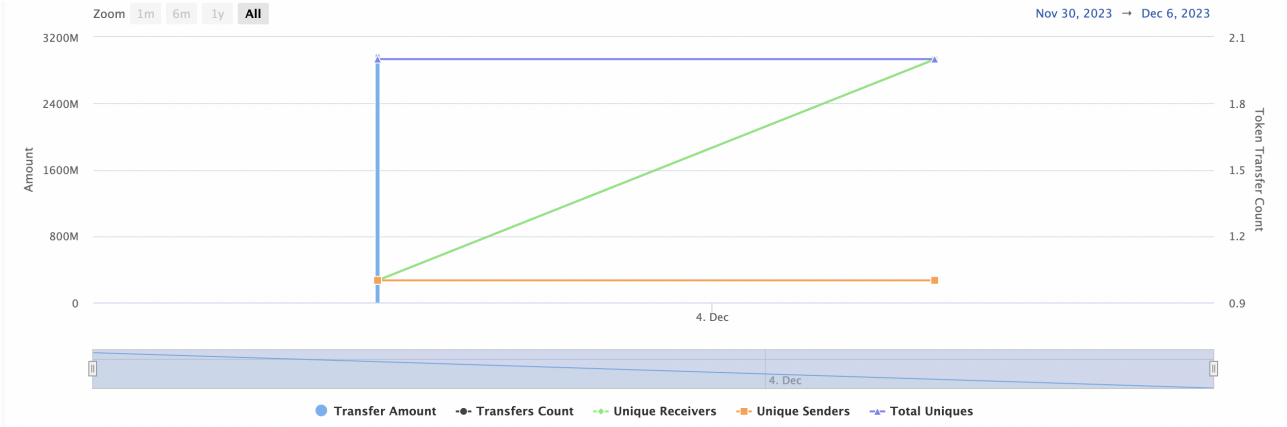
Project Twitter: **https://twitter.com/hegs_official_**

Project Telegram: **https://t.me/CYRUSOffical**

Project Discord: **https://discord.gg/yPGRAPkUcE**

Project Twitch: **https://www.twitch.tv/degenmillionairesclub**

Token Analytics



Project Website Overview

✓JavaScript errors hasn’t been found.
✓Malware pop-up windows hasn’t been detected.
✓No issues with loading elements, code, or stylesheets.

Vulnerabilities checking

Issue Description	Checking Status
Compiler Errors	Completed
Delays in Data Delivery	Completed
Re-entrancy	Completed
Transaction-Ordering Dependence	Completed
Timestamp Dependence	Completed
Shadowing State Variables	Completed
DoS with Failed Call	Completed
DoS with Block Gas Limit	Completed
Outdated Complier Version	Completed
Assert Violation	Completed
Use of Deprecated Solidity Functions	Completed
Integer Overflow and Underflow	Completed
Function Default Visibility	Completed
Malicious Event Log	Completed
Math Accuracy	Completed
Design Logic	Completed
Fallback Function Security	Completed
Cross-function Race Conditions	Completed
Safe Zeppelin Module	Completed

Security Issues

1) Block timestamp:

Check: timestamp

Severity: Low

Confidence: **Medium**

L376-396, L399-409, L436-L458

```
function releaseLock(address _holder↑) internal whenNotPaused {
    uint256 totalReleased = 0;
    LockInfo[] storage locks = lockInfo[_holder↑];

    for (uint256 i = 0; i < locks.length;) {
        if (locks[i].releaseTime <= block.timestamp) {
            totalReleased += locks[i].balance;

            locks[i] = locks[locks.length - 1];
            locks.pop();
        } else {
            i++;
        }
    }

    if (totalReleased > 0) {
        _balances[_holder↑] += totalReleased;
        totalLocked[_holder↑] -= totalReleased;
        emit Unlock(_holder↑, totalReleased);
    }
}
```

```
function lock(address _holder↑, uint256 _amount↑, uint256 _releaseTime↑) public onlyOwner whenNotPaused {
    require(_balances[_holder↑] >= _amount↑, "Balance is too small.");
    require(block.timestamp <= _releaseTime↑, "TokenTimelock: release time is before current time");

    _balances[_holder↑] -= _amount↑;
    totalLocked[_holder↑] += _amount↑;  // 잠금된 토큰의 양 갱신
    lockInfo[_holder↑].push(LockInfo(_releaseTime↑, _amount↑));

    emit Transfer(_holder↑, address(0), _amount↑);  // 잠금을 반영하는 Transfer 이벤트 발생
    emit Lock(_holder↑, _amount↑, _releaseTime↑);
}
```

```
function releaseMyExpiredLocks() public whenNotPaused {
    uint256 totalReleased = 0;
    LockInfo[] storage locks = lockInfo[msg.sender];
    uint256 length = locks.length;

    for (uint256 i = 0; i < length;) {
        if (locks[i].releaseTime <= block.timestamp) {
            totalReleased += locks[i].balance;

            locks[i] = locks[length - 1];
            locks.pop();
            length--;
        } else {
            i++;
        }
    }

    if (totalReleased > 0) {
        _balances[msg.sender] += totalReleased;
        totalLocked[msg.sender] -= totalReleased;
        emit Unlock(msg.sender, totalReleased);
    }
}
```

Description: Dangerous usage of block.timestamp. block.timestamp can be manipulated by miners.

Recommendation: Avoid relying on block. timestamp.

2) Conformance to Solidity naming conventions:

Check: naming conventions

Severity: Low

Confidence: **High**

L330, L338, L348, L371, L376, L399, L428, L461, L466, L483

```
function transferOwnership(address _newOwner↑) public onlyOwner whenNotPaused
    require(_newOwner↑ != address(0), "New owner cannot be the zero address");
    require(_newOwner↑ != owner, "New owner cannot be the current owner");
    emit OwnershipTransferred(owner, _newOwner↑);
    owner = _newOwner↑;
}
```

```
function transferFrom(address _from↑, address _to↑, uint256 _value↑) public override whenNotPaused returns (bool) {
```

```
function balanceOf(address _holder↑) public view override returns (uint256) {
```

```
function releaseLock(address _holder↑) internal whenNotPaused {
```

```
function lockCount(address _holder↑) public view returns (uint256) {
```

```
function transferWithLock(address _to↑, uint256 _value↑, uint256 _releaseTime↑) public onlyOwner whenNotPaused returns (bool) {
```

```
function lockTimeChange(address _holder↑, uint256 _idx↑, uint256 _releaseTime↑) public onlyOwner whenNotPaused {
```

```
function lock(address _holder↑, uint256 _amount↑, uint256 _releaseTime↑) public onlyOwner whenNotPaused {
```

Description: solidity defines a naming convention that should be followed.

Recommendation: it is recommended to follow the solidity naming convention.

3) Dead-Code:

Check: dead code

Severity: Informational

Confidence: **Medium**

L 261-266

```
function _burnFrom(address owner↑, uint256 amount↑) internal {
    _burn(owner↑, amount↑);
    uint256 currentAllowance = _allowances[owner↑][msg.sender];
    require(currentAllowance >= amount↑, "ERC20: burn amount exceeds allowance");
    _approve(owner↑, msg.sender, currentAllowance - amount↑);
}
```

Description: a function that is not sued.

Recommendation: it is recommended to remove unused functions.

Conclusion for project owner

Medium and Low-severity issues exist within smart contracts.

NOTE: Please check the disclaimer above and note, that the audit makes no statements or warranties on the business model, investment attractiveness, or code sustainability. Contract security report for community

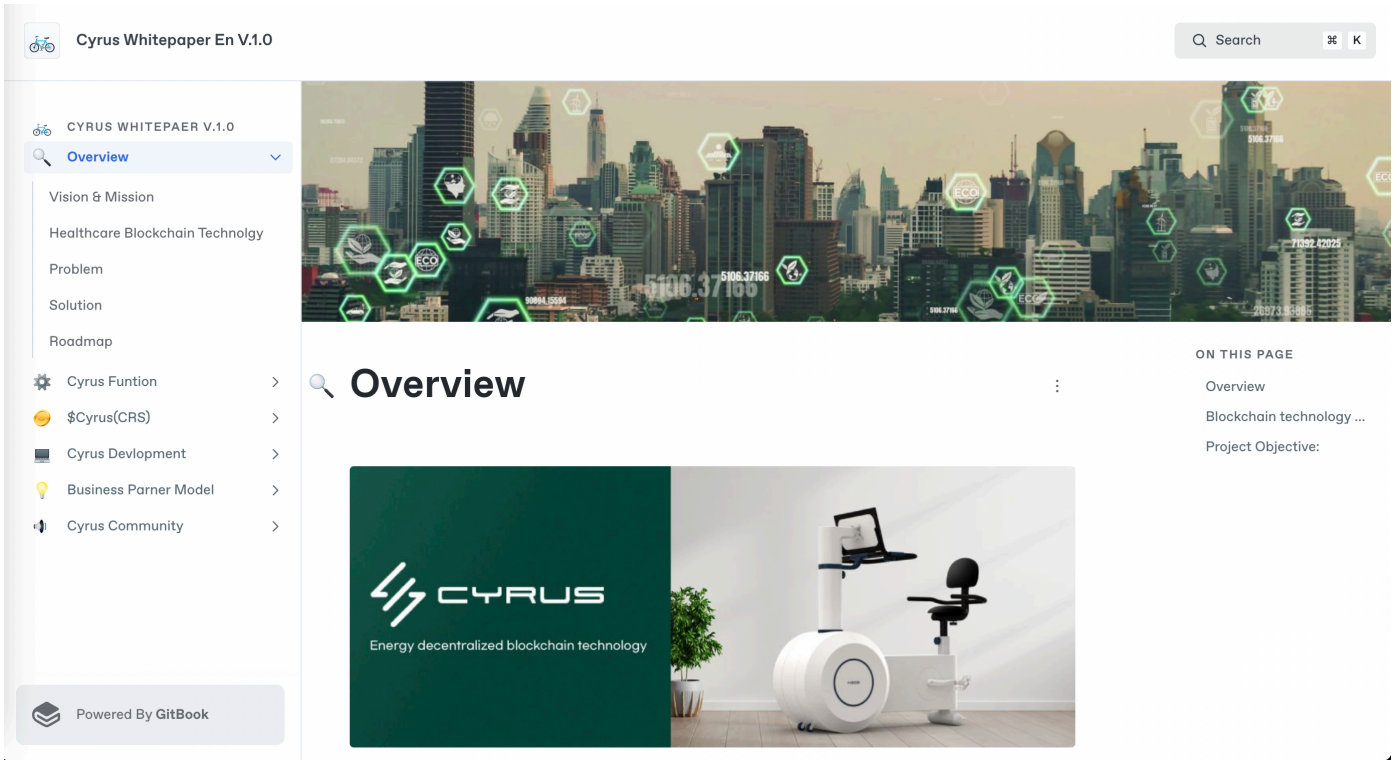
SECURITY REPORT
FOR COMMUNITY

CYRUS

Approved

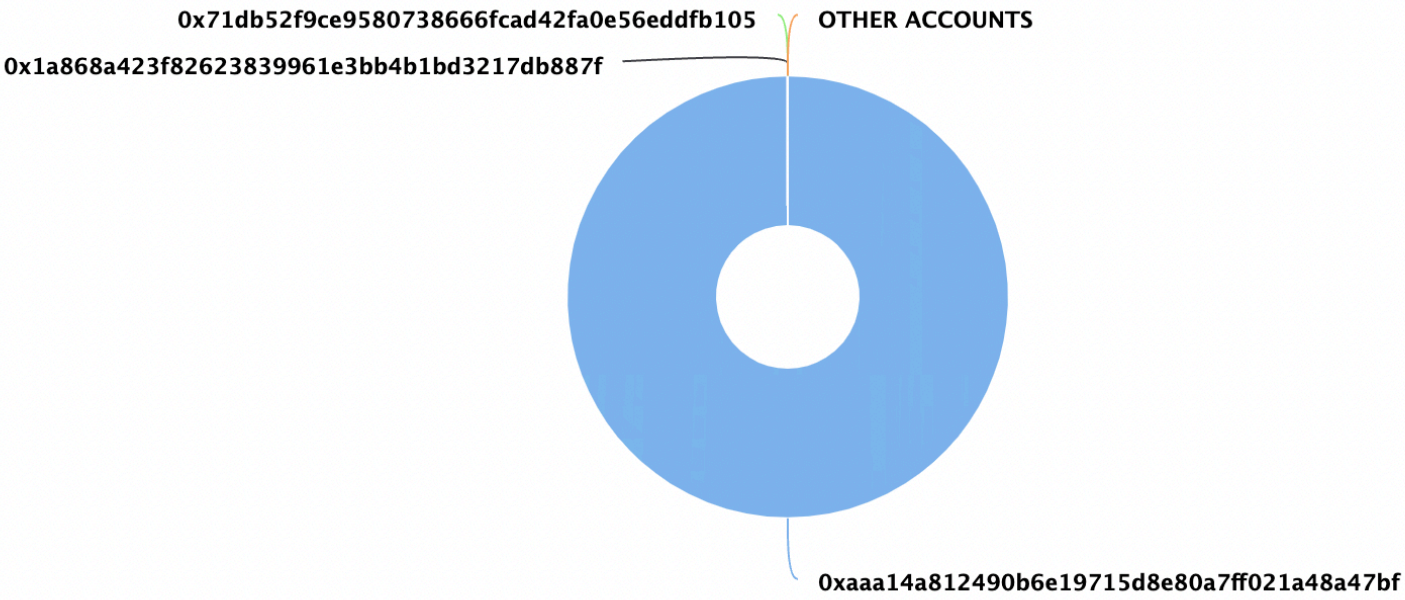
Whitepaper of the project

The whitepaper of CYRUS project has been verified on behalf of Approved team.



Whitepaper link: **https://organization-nhl.gitbook.io/copy-of-sylas-whitepaper-en-v.1.0/cyrus-whitepaer-v.1.0/overview**

CRS Token Distribution



CRS Top 10 Holders

Rank	Address	Quantity (Token)	Percentage
1	0xaAA14a812490b6e19715D8E80a7Ff021a48A47bf	2,999,999,900	100.0000%
2	0x1a868A423f82623839961E3bB4B1bD3217DB887f	50	0.0000%
3	0x71db52F9ce9580738666FCaD42Fa0e56eDDFB105	50	0.0000%

Website: https://approved.ltd

Telegram: @team_approved

GitHub: https://github.com/Approved-Audits/smart_contracts