Gold

COGNITOS

The next generation of blockchain security

Penguin WAK

Gold Audit

Deep Scan Mode Screening

March, 12
2023

Disclaimer

Cognitos provides due-diligence project audits for various projects. Cognitos in no way guarantees that a project will not remove liquidity, sell off teamsupply, or otherwise exit scam.

Cognitos does the legwork and provides public information about the project in an easy-to-understand format for the common person.

Agreeing to an audit in no way guarantees that a team will not remove all liquidity ("Rug Pull"), remove liquidity slowly, sell off tokens, quit the project, or completely exit scam. There is also no way to prevent private sale holders from selling off their tokens. It is ultimately your responsibility to read through all documentation, social media posts, and contract code of each individual project to draw your own conclusions and set your own risk tolerance.

Cognitos in no way takes responsibility for any losses, nor does Cognitos encourage any speculative investments. The information provided in this audit is for information purposes only and should not be considered investment advice. Cognitos does not endorse, recommend, support, or suggest any projects that have been audited. An audit is an informational report based on our findings, We BEP recommend you do your own research, we will never endorse any project to invest in.

The badge of Audit, KYC, Vetted and Safu is not a guarantee for safety. your reliance on a badge is solely at your own risk. we are not responsible for your investment loss and hereby expressly disclaim any liabilities that may arise from your use or reference of the badge.

Table of content

Audit Scope

Cognitos was comissioned by Penguin wak to perform an audit based on the following code:

https://bscscan.com/token/0x0b6d7735E0430D48675cba2955E87ccb0cD754cF#code

Note that we only audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

Audit Method
Cognitos's manual smart contract audit is an extensive methodical examination and analysis of the smart contract's code that is used to interact with the blockchain. This process is conducted to discover errors, issues and security vulnerabilities in the code in order to suggest improvements and ways to fix them.

Automated Vulnerability Check
Cognitos uses software that checks for common vulnerability issues within smart contracts. We use automated tools that scan the contract for security vulnerabilities such as integer-overflow, integer-underflow, out-of-gas-situations, unchecked transfers, etc.

Manual Code Review
Cognitos's manual code review involves a human looking at source code, line by line, to find vulnerabilities. Manual code review helps to clarify the context of coding decisions. Automated tools are faster but they cannot take the developer's intentions and general business logic into consideration.

Project Overview

Name & Logo

Penguin WAK

Project Statement

Penguin innovations utilize blockchain technology alongside in-depth knowledge from its team to proffer solutions for effective management of finances in the crypto space. Backed by Decentralization and fully fledged DAO governance structure.

Website & Social Media

• Website	https://penguintoken.site/
• Telegram	https://t.me/penguinwakk
• Discord	-
• Twitter	https://twitter.com/penguintoken2
• Medium	https://medium.com/@penguinwak
• Instagram	-
• Youtube	-

Blockchain

• Network	Binance Smart Chain
• Contract	0x0b6d7735E0430D48675cba2955E87ccb0cD754cF (verified)

Token Data

Token Symbol	WAK
Token Name	Penguin WAK
Contract Address	0x0b6d7735E0430D48675cba2955E87ccb0cD754cF
Compiler Version	v0.8.17+commit.8df45f5f
Total Supply	100,000,000,000 WAK
Decimals	18
Contract Creator	0x1012732040842aff8b7f8ef8767a4e7fe72da1bc
Contract Owner	0x8befb5fb7fa2db92a1e043559486d31c24bb648a

Security Detection

Risky Item 2

Attention Item 3

Safe Attention Risky

Contract Security

Contract Verified	Yes
Proxy Contract	No
Mint Function	Yes
Retrieves Ownership Function	No
Authority to Change Balance	Yes
Hidden Owner	No
Self-destruct Function	No
External Call Risk	No

Honeypot Risk

Appear to be a Honeypot	No
Suspend Trading Function	Yes
Can Sell all of the Token	Yes
Can be Bought	Yes
Trading Cooldown Function	No
Anti_whale Function	Yes
Tax Modified Function	Yes
Blacklist Function	Yes
Whitelist Function	No
Personal Addresses Tax Changes	No

Vulnerability Summary

Total Findings

- 0 Critical
- 1 Major
- 5 Medium
- 5 Minor
- 7 Informational

Severity

- Critical

-

- Major

Use Of Tx.origin

- Medium

Incorrect Access Control
Unchecked Array Length
Deleting A Mapping Within A Struct
Incorrect Shift Assembly
Approve Front-running Attack

- Minor

Internal Functions Never Used
Outdated Compiler Version
Use Of Floating Pragma
Long Number Literals
Missing Events

- Info

Hard-coded Address Detected
Unused Receive Fallback
Missing Indexed Keywords In Events
In-line Assembly Detected
Require With Empty Message
Block Values As A Proxy For Time
Presence Of Overpowered Role

Vulnerability Scan

USE OF TX.ORIGIN

Severity	High
Confidence Parameter	Firm

Vulnerability Description

In Solidity, **tx.origin** is a global variable that returns the address of the account that sent the transaction. Using the variable for authorization could make a contract vulnerable. For example, if an authorized account calls a malicious contract which triggers it to call the vulnerable contract that passes an authorization check since **tx.origin** returns the original sender of the transaction which in this case is the authorized account.

Scanning Line:

```
1497    emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas, tx.origin);
```

```
1633        emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, true, gas, tx.origin);
```

Recommen-dation:

tx.origin should not be used for authorization in smart contracts. It does have some legitimate use cases, for example, To prevent external contracts from calling the current contract, you can implement a require of the form **require(tx.origin == msg.sender)**. This prevents intermediate contracts from calling the current contract, thus limiting the contract to regular codeless addresses.

Incorrect Access Control

Severity	Medium
Confidence Parameter	Firm

Vulnerability Description

Access control plays an important role in segregation of privileges in smart contracts and other applications. If this is misconfigured or not properly validated on sensitive functions, it may lead to loss of funds, tokens and in some cases compromise of the smart contract.

The contract **IUniswapV2Pair** is importing an access control library @open-zeppelin/contracts/access/AccessControl.sol but the function **burn** is missing the modifier **onlyRole**.

Scanning Line:

```
.730    function burn(address to) external returns (uint amount0, uint amount1);
```

The contract **DividendPayingToken** is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **withdrawDividend** is missing the modifier **onlyOwner**.

Scanning Line:

```
836  function withdrawDividend() public virtual override {
837    _withdrawDividendOfUser(payable(msg.sender));
838  }
```

The contract **TokenDividendTracker** is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **process** is missing the modifier **onlyOwner**.

```
1102    function process(uint256 gas) public returns (uint256, uint256, uint256) {
1103        uint256 numberOfTokenHolders = tokenHoldersMap.keys.length;
1104
1105        if(numberOfTokenHolders == 0) {
1106            return (0, 0, lastProcessedIndex);
1107        }
```

```

    uint256 _lastProcessedIndex = lastProcessedIndex;

   uint256 gasUsed = 0;

    uint256 gasLeft = gasleft();

    uint256 iterations = 0;
    uint256 claims = 0;

    while(gasUsed < gas && iterations < numberOfTokenHolders) {
      _lastProcessedIndex++;

      if(_lastProcessedIndex >= tokenHoldersMap.keys.length) {
          _lastProcessedIndex = 0;
      }

      address account = tokenHoldersMap.keys[_lastProcessedIndex];

      if(canAutoClaim(lastClaimTimes[account])) {
          if(processAccount(payable(account), true)) {
              claims++;
          }
      }

      iterations++;

      uint256 newGasLeft = gasleft();

      if(gasLeft > newGasLeft) {
          gasUsed = gasUsed.add(gasLeft.sub(newGasLeft));
      }

      gasLeft = newGasLeft;
    }

    lastProcessedIndex = _lastProcessedIndex;

    return (iterations, claims, lastProcessedIndex);
}
```

The contract **PenguinWAK** is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **claim** is missing the modifier **onlyOwner**.

Scanning Line:

```
function claim() external {
  dividendTracker.processAccount(payable(msg.sender), false);
}
```

The contract **TokenDividendTracker** is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **MAPRemove** is missing the modifier **onlyOwner**.

Scanning Line:

```
function MAPRemove(address key) public {
  if (!tokenHoldersMap.inserted[key]) {
    return;
  }

  delete tokenHoldersMap.inserted[key];
  delete tokenHoldersMap.values[key];

  uint index = tokenHoldersMap.indexOf[key];
  uint lastIndex = tokenHoldersMap.keys.length - 1;
  address lastKey = tokenHoldersMap.keys[lastIndex];

  tokenHoldersMap.indexOf[lastKey] = index;
  delete tokenHoldersMap.indexOf[key];

  tokenHoldersMap.keys[index] = lastKey;
  tokenHoldersMap.keys.pop();
}
```

The contract **TokenDividendTracker** is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **MAPSet** is missing the modifier **onlyOwner**.

Scanning Line:

```
function MAPSet(address key, uint val) public {
    if (tokenHoldersMap.inserted[key]) {
        tokenHoldersMap.values[key] = val;
    } else {
        tokenHoldersMap.inserted[key] = true;
        tokenHoldersMap.values[key] = val;
        tokenHoldersMap.indexOf[key] = tokenHoldersMap.keys.length;
        tokenHoldersMap.keys.push(key);
    }
}
```

The contract **PenguinWAK** is importing an access control library @openzeppelin/contracts/access/Ownable.sol but the function **processDividendTracker** is missing the modifier **onlyOwner**.

Scanning Line:

```
function processDividendTracker(uint256 gas) external {
    (uint256 iterations, uint256 claims, uint256 lastProcessedIndex) = dividendTracker.process(gas);
    emit ProcessedDividendTracker(iterations, claims, lastProcessedIndex, false, gas, tx.origin);
}
```

Recommendation:

It is recommended to go through the contract and observe the functions that are lacking an access control modifier. If they contain sensitive administrative actions, it is advised to add a suitable modifier to the same

Unchecked Array Length

Severity	Medium
Confidence Parameter	Tentative

Vulnerability Description

Ethereum is a very resource-constrained environment. Prices per computational step are orders of magnitude higher than with centralized providers. Moreover, Ethereum miners impose a limit on the total number of Gas consumed in a block. If **array.length** is large enough, the function exceeds the block gas limit, and transactions calling it will never be confirmed.

for (uint256 i = 0; i < array.length ; i++) { cosltyFunc(); }

This becomes a security issue, if an external actor influences **array.length**. E.g., if an array enumerates all registered addresses, an adversary can register many addresses, causing the problem described above.

Scanning Line:

```
1377    for(uint256 i = 0; i < accounts.length; i++) {
```

Recommendation:

Either explicitly or just due to normal operation, the number of iterations in a loop can grow beyond the block gas limit, which can cause the complete contract to be stalled at a certain point. Therefore, loops with a bigger or unknown number of steps should always be avoided.

Deleting A Mapping Within A Struct

Severity	Medium
Confidence Parameter	Tentative

Vulnerability Description

The contract was found to be using a mapping (X) containing a struct (Y). This struct also contains another mapping (Z).
The vulnerability arises when the an item is deleted from mapping (X). This does not delete data from mapping (Z). The remaining data may compromise the contract

Scanning Line:

```
1194    delete tokenHoldersMap.inserted[key];
1195    delete tokenHoldersMap.values[key];
1202    delete tokenHoldersMap.indexOf[key];
```

Recommendation:

A lock mechanism should be implemented instead of deletion to disable the struct containing the mapping.

Incorrect Shift In Assembly

Severity	Medium
Confidence Parameter	Tentative

Vulnerability Description

Assembly usage in smart contracts should be done with utmost case as these statements bypass certain security checks and most of the times are more difficult to implement than a normal solidity code.
The shift statement in assembly (**shr, sar,** or **shl**) uses two parameters inside itself. The first argument defines the number of shifts and the second one defines the parameter on which the shift is to happen. These values should not be reversed as this will change the logic of the contract.
The contract is using **shl(x,y)** inside the assembly.

Scanning Line:

```
214    mstore(add(ptr, 0x14), shl(0x60, implementation))
232    mstore(add(ptr, 0x14), shl(0x60, implementation))
252    mstore(add(ptr, 0x38), shl(0x60, deployer))
```

Recommendation:

It is recommended to go through the assembly usage in the code to make sure that the parameters passed in the shift operations used in the contract are in the correct order.

Weakness Classification

		AI Scan	Human Review	Result
CTS 000	Function Default Visibility	✓	✓	Passed
CTS 001	Integer Overflow and Underflow	✓	✓	Passed
CTS 002	Outdated Compiler Version	✓	✓	Passed
CTS 003	Floating Pragma	LOW	✓	Passed
CTS 004	Unchecked Call Return Value	✓	✓	Passed
CTS 005	Unprotected Ether Withdrawal	✓	✓	Passed
CTS 006	Unprotected SELFDESTRUCT Instruction	✓	✓	Passed
CTS 007	Reentrancy	✓	✓	Passed
CTS 008	State Variable Default Visibility	✓	✓	Passed
CTS 009	Uninitialized Storage Pointer	✓	✓	Passed
CTS 010	Assert Violation	✓	✓	Passed
CTS 011	Use of Deprecated Solidity Functions	✓	✓	Passed
CTS 012	Delegatecall to Untrusted Callee	✓	✓	Passed
CTS 013	DoS with Failed Call	✓	✓	Passed
CTS 014	Transaction Order Dependence	✓	✓	Passed
CTS 015	Authorization through tx.origin	✓	✓	Passed
CTS 016	Block values as a proxy for time	✓	✓	Passed
CTS 017	Signature Malleability	✓	✓	Passed
CTS 018	Incorrect Constructor Name	✓	✓	Passed

		AI Scan	Human Review	Result
CTS 019	Shadowing State Variables	✓	✓	Passed
CTS 020	Weak Sources of Randomness from Chain Attributes	✓	✓	Passed
CTS 021	Missing Protection against Signature Replay Attacks	✓	✓	Passed
CTS 022	Lack of Proper Signature Verification	✓	✓	Passed
CTS 023	Requirement Violation	✓	✓	Passed
CTS 024	Write to Arbitrary Storage Location	✓	✓	Passed
CTS 025	Incorrect Inheritance Order	✓	✓	Passed
CTS 026	Insufficient Gas Griefing	✓	✓	Passed
CTS 027	Arbitrary Jump with Function Type Variable	✓	✓	Passed
CTS 028	DoS With Block Gas Limit	✓	✓	Passed
CTS 029	Typographical Error	✓	✓	Passed
CTS 030	Right-To-Left-Override control character (U+202E)	✓	✓	Passed
CTS 031	Presence of unused variables	✓	✓	Passed
CTS 032	Unexpected Ether balance	✓	✓	Passed
CTS 033	Hash Collisions With Multiple Variable Length Arguments	✓	✓	Passed
CTS 034	Message call with hardcoded gas amount	✓	✓	Passed
CTS 035	Code With No Effects	✓	✓	Passed
CTS 036	Unencrypted Private Data On-Chain	✓	✓	Passed

Website Security

Security Detection

Our automated scan did not detect malware on your site.

Sitescan Report

Normalized URL	http://penguintoken.site:80
Submission date	Sun Mar 12 10:41:14 2023
Server IP address	199.188.200.245
Country	United States
Web Server	LiteSpeed
Malicious files	0
Suspicious files	0
Potentially Suspicious files	0
Clean files	91
External links detected	265
Iframes scanned	21
Blacklisted	No

Scanned files analysis

Malicious files	0
Suspicious files	0
Potentially Suspicious files	0
Clean files	91

Malware Checked

- No malware detected by scan (Low Risk)
- No injected spam detected (Low Risk)
- No defacements detected (Low Risk)
- No internal server errors detected (Low Risk)

Blacklist Checked

- Domain clean by Google Safe Browsing
- Domain clean by McAfee
- Domain clean by Sucuri Labs
- Domain clean by ESET
- Domain clean by PhishTank
- Domain clean by Yandex
- Domain clean by Opera

SSL Checked

- penguintoken.site resolves to 199.188.200.245
- The certificate should be trusted by all major web browsers
- The certificate was issued by Sectigo.
- The certificate will expire in 117 days.
- The hostname (penguintoken.site) is correctly listed in the certificate.

Server

Common name: penguintoken.site
SANs: penguintoken.site, www.penguintoken.site
Valid from July 6, 2022 to July 7, 2023
Serial Number: 7ebd21f07840722679765809d10e2c4e
Signature Algorithm: sha256WithRSAEncryption
Issuer: Sectigo RSA Domain Validation Secure Server CA

Chain 1

Common name: Sectigo RSA Domain Validation Secure Server CA
Organization: Sectigo Limited
Location: Salford, Greater Manchester, GB
Valid from November 1, 2018 to December 31, 2030
Serial Number: 7d5b5126b476ba11db74160bbc530da7
Signature Algorithm: sha384WithRSAEncryption
Issuer: USERTrust RSA Certification Authority

Chain 2

Common name: USERTrust RSA Certification Authority
Organization: The USERTRUST Network
Location: Jersey City, New Jersey, US
Valid from March 11, 2019 to December 31, 2028
Serial Number: 3972443af922b751d7d36c10dd313595
Signature Algorithm: sha384WithRSAEncryption
Issuer: AAA Certificate Services

Technology Profiler

CMS	WordPress 6.1.1
Blogs	WordPress 6.1.1
Miscellaneous	Webpack 50% sure Module Federation 50% sure
Web servers	LiteSpeed
Programming languages	PHP
Databases	MySQL
Page builder	Elementor 3.11.3
WordPress plugins	Elementor 3.11.3 WP-Optimize
Performance	WP-Optimize

Team Data

Dev & Team Informations

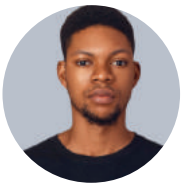
We found developer and team information on the website

Soud Altukhaim
PenguinWak Founder
linkedin.com/in/soud-altukhaim-b652bab9/
t.me/penguinwakt

Piyanut Wonglakhon
Blockchain Advisor
linkedin.com/in/piyanut-wonglakhon-a353171a6/
penguintoken.site/wp-content/uploads/2023/02/Piyanut-Wonglakhon-BIO.html

Denis Lukavackic
Website Developer
linkedin.com/in/denis-lukavackic-1b3338106/
penguintoken.site/wp-content/uploads/2023/02/Denis-Lukavackic-BIO.html

Damir Lukavačkić
Marketing Advisor
linkedin.com/in/damir-lukavackic-057828240/
penguintoken.site/wp-content/uploads/2023/02/Damir-Lukavackic-BIO.html



Dominic Gregory
Research Manager
linkedin.com/in/dominic-agboh-63b5841a9/
twitter.com/GregoryOkwudili
penguintoken.site/wp-content/uploads/2023/02/Dominics-CV.pdf

Agbaje Mubarak
Project Designer
twitter.com/agbaje_mubarak
t.me/agbajeTech

Piyachai Wonglakon
Token Advisor
linkedin.com/in/piyachai-wonglakon-44a3071a6/
penguintoken.site/wp-content/uploads/2023/02/PIYACHAI-WONGLAKON-BIO.html

Damir Hajdič
Community Manager
linkedin.com/in/damir-hajdi%C4%87-93548918b/
penguintoken.site/wp-content/uploads/2023/02/Damir-Hajdic-ALL-INFO-with-links.html

Cognitos Project Audit has been completed for **Penguin WAK - BSC**

Block number : 0000078

This result is only valid if viewed on www.cognitos.io