Cyberscope

# Audit Report

# OpenGames

September 2023

SHA256 910ff3ea1d9144b290281b34e7b0baf2e9b6ea3f73ff9b9c1361a0f6ac62f1d5

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 |
|---|---|---|---|
| | MU | Modifiers Usage | Unresolved |
| | RPF | Redundant Pausable Functionality | Unresolved |
| | RFD | Redundant Function Declarations | Unresolved |
| | UM | Unused Modifier | Unresolved |
| | CO | Contract Optimization | Unresolved |
| | MNF | Misleading NFT Functionality | Unresolved |
| | RSML | Redundant SafeMath Library | Unresolved |
| | L17 | Usage of Solidity Assembly | Unresolved |
| | L19 | Stable Compiler Version | Unresolved |

# Table of Contents

# Review

| | |
|---|---|
| **Contract Name** | OpenGamesBuildersToken |
| **Testing Deploy** | https://testnet.bscscan.com/address/0x75628deec8489c5cdc0abe5ddf46b73b35214a1c |
| **Symbol** | OGBX |
| **Decimals** | 18 |
| **Total Supply** | 1,000,000,000 |

## Audit Updates

| | |
|---|---|
| **Initial Audit** | 10 Feb 2023 https://github.com/cyberscope-io/audits/blob/main/opengames/v1/audit.pdf |
| **Corrected Phase 2** | 13 Feb 2023 https://github.com/cyberscope-io/audits/blob/main/opengames/v2/audit.pdf |
| **Corrected Phase 3** | 14 Sep 2023 |

## Source Files

| Filename | SHA256 |
| --- | --- |
| **contracts/OGBToken.sol** | 910ff3ea1d9144b290281b34e7b0baf2e9b6ea3f73ff9b9c1361a0f6ac62f1d5 |
| **@openzeppelin/contracts/utils/Context.sol** | 1458c260d010a08e4c20a4a517882259a23a4baa0b5bd9add9fb6d6a1549814a |
| **@openzeppelin/contracts/utils/math/SafeMath.sol** | fc16aa4564878e1bb65740239d0c1422451cd32136306626ac37f5d5e0606a7b |
| **@openzeppelin/contracts/token/ERC20/IERC20.sol** | 7ebde70853ccafcf1876900dad458f46eb9444d591d39bfc58e952e2582f5587 |
| **@openzeppelin/contracts/token/ERC20/ERC20.sol** | d20d52b4be98738b8aa52b5bb0f88943f62128969b33d654fbca731539a7fe0a |
| **@openzeppelin/contracts/token/ERC20/extensions/IERC20Metadata.sol** | af5c8a77965cc82c33b7ff844deb9826166689e55dc037a7f2f790d057811990 |
| **@openzeppelin/contracts/security/Pausable.sol** | 2072248d2f79e661c149fd6a6593a8a3f038466557c9b75e50e0b001bcb5cf97 |
| **@openzeppelin/contracts/access/Ownable.sol** | a8e4e1ae19d9bd3e8b0a6d46577eec098c01fbaffd3ec1252fd20d799e73393b |

# Findings Breakdown


| | |
|---|---|
| Critical | 0 |
| Medium | 0 |
| Minor / Informative | 9 |

| Severity | Unresolved | Acknowledged | Resolved | Other |
|---|---|---|---|---|
| Critical | 0 | 0 | 0 | 0 |
| Medium | 0 | 0 | 0 | 0 |
| Minor / Informative | 9 | 0 | 0 | 0 |

# MU - Modifiers Usage

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.sol#L56,68,77,86,97,106 |
| **Status** | Unresolved |

## Description

The contract uses 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(amount > 0, "Amount must be greater than 0");
```

## 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.

## RPF - Redundant Pausable Functionality

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.sol#L43,48 |
| **Status** | Unresolved |

### Description

The contract inherits functionality from the Pausable contract, which provides features for pausing and unpausing the contract's operations. However, none of the contract's functions make use of the `whenNotPaused` and `whenPaused` modifiers inherited from Pausable. Consequently, these modifiers remain unused and redundant within the contract.

```
// Function to pause contract operations
function pause() external onlyOwner {
    _pause();
}

// Function to resume contract operations
function unpause() external onlyOwner {
    _unpause();
}
```

### Recommendation

To optimize the contract code and eliminate redundancy, it is advisable to either remove the inheritance from the Pausable contract if pausing and unpausing functionality is not needed, or to consider integrating these modifiers into the contract's functions where appropriate. If pausing functionality is indeed required, ensure that the `whenNotPaused` and `whenPaused` modifiers are applied to relevant functions to enable the contract's pausing mechanism effectively. This way, the contract remains more coherent and efficient in its operation.

# RFD - Redundant Function Declarations

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.sol#L53,65 |
| **Status** | Unresolved |

## Description

The contract contains a set of custom transfer functions ( `inGameTransaction` , `upgradeItems` , `payMarketplaceFees` , `buyNFT` , and `sellNFT` ) that essentially replicate the functionality of the standard ERC20 transfer function with additional validation. These custom functions share a common pattern and differ primarily in their names and validation checks. As a result, declaring all these functions is redundant.

```
// Function for in-game transactions
    function inGameTransaction(address to, uint256 amount) external {
    // This function allows users to perform in-game transactions by
transferring tokens to another address.
    // The "to" parameter is the recipient's address, and "amount" is the
number of tokens to transfer.
    require(amount > 0, "Amount must be greater than 0");
    require(to != address(0), "Invalid recipient address");
    // The require statement checks that the transfer amount is positive.
    // If it's not, the function will revert with the provided error
message.
    _transfer(msg.sender, to, amount);
    // The _transfer function, inherited from ERC20, is used to transfer
tokens from the sender to the recipient.
}
...
```

## Recommendation

To streamline the contract code and adhere to best practices, it is recommended to leverage the standard ERC20 transfer function and implement custom logic using the `_beforeTokenTransfer` function provided by the OpenZeppelin ERC20 contract (or similar functionality). By centralizing the transfer logic in a single place, you can reduce redundancy, improve code maintainability, and ensure consistent validation checks across all transfer operations. Custom behavior, such as additional validation or functionality, can be implemented within `_beforeTokenTransfer` as needed, allowing for more modular and efficient code while retaining flexibility for customization.

# UM - Unused Modifier

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.sol#L87,98,107 |
| **Status** | Unresolved |

## Description

The contract defines a `hasMinimumBalance` modifier intended to ensure that the caller has a minimum balance of tokens before executing certain functions. However, instead of consistently applying this modifier where needed, the contract redundantly contains direct `require` statements within the functions, performing the same balance check as the modifier.

```
// Modifier to ensure that the caller has a minimum balance of tokens
modifier hasMinimumBalance(uint256 amount) {
    require(balanceOf(msg.sender) >= amount, "Insufficient balance");
    _;
}

require(balanceOf(msg.sender) >= nftPrice, "Insufficient balance");
```

## Recommendation

To promote code consistency, readability, and maintenance, it is recommended to use the `hasMinimumBalance` modifier consistently where balance checks are required. By doing so, you can eliminate redundancy, reduce the likelihood of errors, and make the codebase more coherent. Review and update all instances where direct `require` statements for balance checks exist within functions, replacing them with the `hasMinimumBalance` modifier. This ensures that the balance check logic remains centralized within the modifier, enhancing code quality and reducing duplication.

# CO - Contract Optimization

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **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 `balanceOfUser` function is redundant. Users can utilize the already defined `balanceOf` function.

```
// Function to retrieve user's token balance
function balanceOfUser(address user) external view returns (uint256) {
    return balanceOf(user);
}
```

## 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.

## MNF - Misleading NFT Functionality

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.sol#L83,94,103 |
| **Status** | Unresolved |

### Description

Several functions within the contract have names that imply they facilitate the buying, selling, or renting of NFTs ( `buyNFT` , `sellNFT` , `rentNFT` ). However, upon closer examination of their implementations, it becomes apparent that these functions are primarily used for transferring tokens rather than handling actual NFTs. This discrepancy between function names and their actual behavior can lead to confusion and misinterpretation.

```
// Function for buying NFTs
function buyNFT(address nftOwner, uint256 nftPrice) external {
    // This function enables users to buy NFTs from the specified owner.
    // The "nftOwner" parameter is the owner's address, and "nftPrice" is
the NFT's price in tokens.
    require(nftPrice > 0, "NFT price must be greater than 0");
    require(balanceOf(msg.sender) >= nftPrice, "Insufficient balance");
    // The require statement checks if the buyer has enough tokens to make
the purchase.
    // If not, the function reverts with an error message.
    _transfer(msg.sender, nftOwner, nftPrice);
}
...
```

## Recommendation

To improve code clarity and ensure that function names accurately represent their behavior, it is recommended to consider renaming these functions to more accurately describe their token transfer functionality. By providing descriptive and accurate function names, you can enhance the readability and understanding of the contract code, reducing the potential for misinterpretation and confusion among developers and users.

## RSML - Redundant SafeMath Library

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.sol |
| **Status** | Unresolved |

### Description

SafeMath is a popular Solidity library that provides a set of functions for performing common arithmetic operations in a way that is resistant to integer overflows and underflows.

Starting with Solidity versions that are greater than or equal to 0.8.0, the arithmetic operations revert to underflow and overflow. As a result, the native functionality of the Solidity operations replaces the SafeMath library. Hence, the usage of the SafeMath library adds complexity, overhead and increases gas consumption unnecessarily.

```
library SafeMath {...}
```

### Recommendation

The team is advised to remove the SafeMath library. Since the version of the contract is greater than `0.8.0` then the pure Solidity arithmetic operations produce the same result.

If the previous functionality is required, then the contract could exploit the `unchecked { ... }` statement.

Read more about the breaking change on https://docs.soliditylang.org/en/v0.8.16/080-breaking-changes.html#solidity-v0-8-0-breaking-changes.

## L17 - Usage of Solidity Assembly

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.sol#L126 |
| **Status** | Unresolved |

### Description

Using assembly can be useful for optimizing code, but it can also be error-prone. It's important to carefully test and debug assembly code to ensure that it is correct and does not contain any errors.

Some common types of errors that can occur when using assembly in Solidity include Syntax, Type, Out-of-bounds, Stack, and Revert.

```
assembly {
            size := extcodesize(addr)
        }
```

### Recommendation

It is recommended to use assembly sparingly and only when necessary, as it can be difficult to read and understand compared to Solidity code.

## L19 - Stable Compiler Version

| | |
|---|---|
| **Criticality** | Minor / Informative |
| **Location** | contracts/OGBToken.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 ^0.8.0;
```

### 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

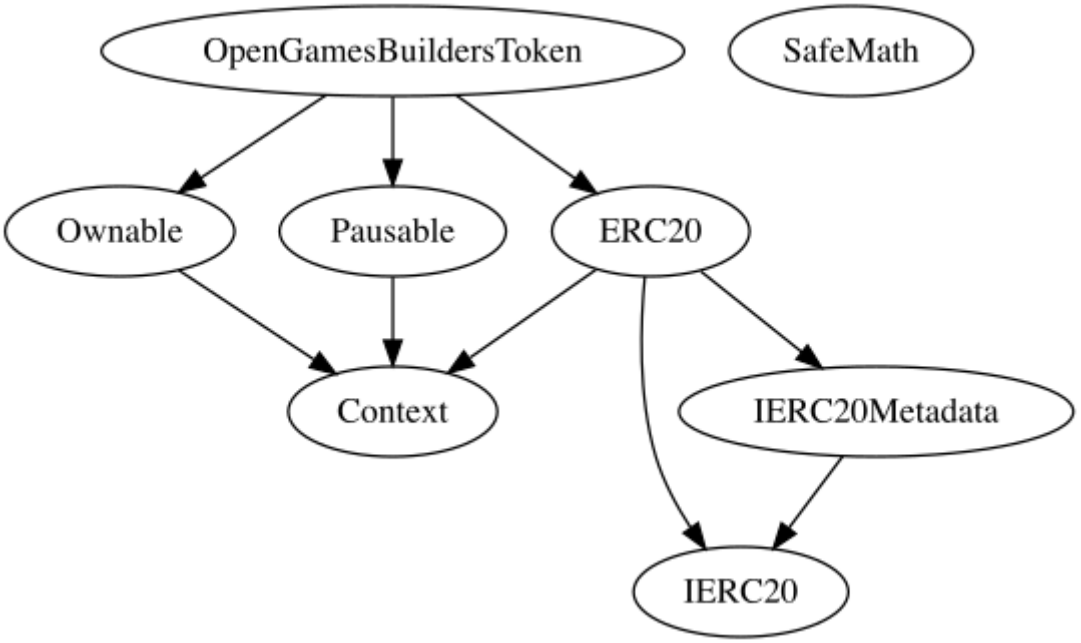
| Contract | Type | Bases | | |
|---|---|---|---|---|
| | **Function Name** | **Visibility** | **Mutability** | **Modifiers** |
| | | | | |
| **OpenGamesBuildersToken** | Implementation | ERC20, Ownable, Pausable | | |
| | | Public | ✓ | ERC20 |
| | pause | External | ✓ | onlyOwner |
| | unpause | External | ✓ | onlyOwner |
| | inGameTransaction | External | ✓ | - |
| | upgradeItems | External | ✓ | - |
| | payMarketplaceFees | External | ✓ | - |
| | buyNFT | External | ✓ | - |
| | sellNFT | External | ✓ | - |
| | rentNFT | External | ✓ | - |
| | autoDiscount | External | | - |
| | isContract | External | | - |
| | contractBalance | External | | - |
| | balanceOfUser | External | | - |
| | withdrawEther | External | ✓ | onlyOwner |
| | | | | |
| **Context** | Implementation | | | |

| | | | | |
|---|---|---|---|---|
| | _msgSender | Internal | | |
| | _msgData | Internal | | |
| | | | | |
| **SafeMath** | Library | | | |
| | tryAdd | Internal | | |
| | trySub | Internal | | |
| | tryMul | Internal | | |
| | tryDiv | Internal | | |
| | tryMod | Internal | | |
| | add | Internal | | |
| | sub | Internal | | |
| | mul | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | sub | Internal | | |
| | div | Internal | | |
| | mod | Internal | | |
| | | | | |
| **IERC20** | Interface | | | |
| | totalSupply | External | | - |
| | balanceOf | External | | - |
| | transfer | External | ✓ | - |
| | allowance | External | | - |

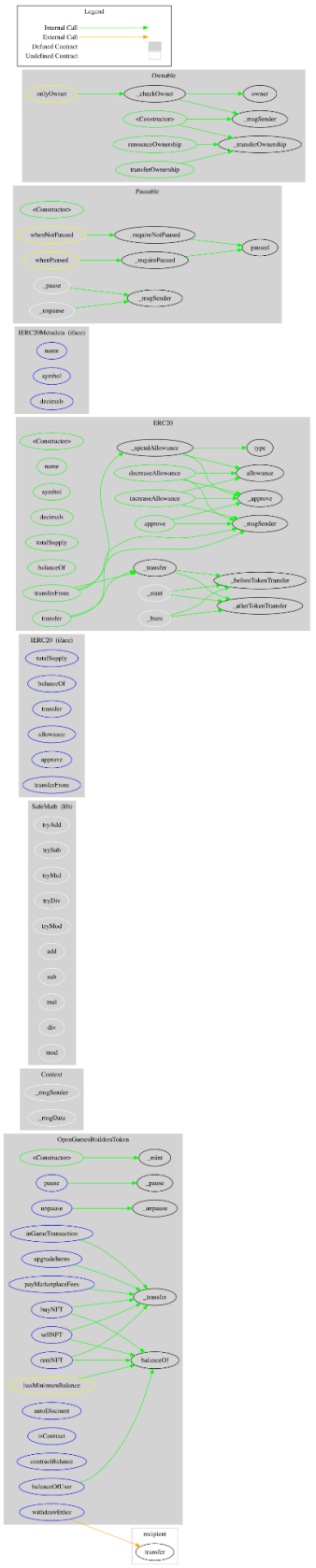
| | | | | |
|---|---|---|---|---|
| | approve | External | ✓ | - |
| | transferFrom | External | ✓ | - |
| | | | | |
| **ERC20** | Implementation | Context, IERC20, IERC20Metadata | | |
| | | Public | ✓ | - |
| | name | Public | | - |
| | symbol | Public | | - |
| | decimals | Public | | - |
| | totalSupply | Public | | - |
| | balanceOf | Public | | - |
| | transfer | Public | ✓ | - |
| | allowance | Public | | - |
| | approve | Public | ✓ | - |
| | transferFrom | Public | ✓ | - |
| | increaseAllowance | Public | ✓ | - |
| | decreaseAllowance | Public | ✓ | - |
| | _transfer | Internal | ✓ | |
| | _mint | Internal | ✓ | |
| | _burn | Internal | ✓ | |
| | _approve | Internal | ✓ | |
| | _spendAllowance | Internal | ✓ | |
| | _beforeTokenTransfer | Internal | ✓ | |

| | | | | |
|---|---|---|---|---|
| | _afterTokenTransfer | Internal | ✓ | |
| | | | | |
| **IERC20Metadata** | Interface | IERC20 | | |
| | name | External | | - |
| | symbol | External | | - |
| | decimals | External | | - |
| | | | | |
| **Pausable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | paused | Public | | - |
| | _requireNotPaused | Internal | | |
| | _requirePaused | Internal | | |
| | _pause | Internal | ✓ | whenNotPaused |
| | _unpause | Internal | ✓ | whenPaused |
| | | | | |
| **Ownable** | Implementation | Context | | |
| | | Public | ✓ | - |
| | owner | Public | | - |
| | _checkOwner | Internal | | |
| | renounceOwnership | Public | ✓ | onlyOwner |
| | transferOwnership | Public | ✓ | onlyOwner |
| | _transferOwnership | Internal | ✓ | |

# Inheritance Graph



# Flow Graph


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
Ownable
onlyOwner
_checkOwner
owner
<Constructor>
_msgSender
renounceOwnership
_transferOwnership
transferOwnership
Pausable
<Constructor>
whenNotPaused
_requireNotPaused
paused
whenPaused
_requirePaused
_pause
_msgSender
_unpause
IERC20Metadata (iface)
name
symbol
decimals
ERC20
<Constructor>
_spendAllowance
type
name
decreaseAllowance
allowance
symbol
increaseAllowance
_approve
decimals
approve
_msgSender
totalSupply
balanceOf
_transfer
transferFrom
_mint
_beforeTokenTransfer
transfer
_burn
_afterTokenTransfer
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
SafeMath (lib)
tryAdd
trySub
tryMul
tryDiv
tryMod
add
sub
mul
div
mod
Context
_msgSender
_msgData
OpenGamesBuildersToken
<Constructor>
_mint
pause
_pause
unpause
_unpause
inGameTransaction
upgradeItems
payMarketplaceFees
_transfer
buyNPT
sellNPT
rentNPT
balanceOf
hasMinimumBalance
autoDiscount
isContract
contractBalance
balanceOfUser
withdrawEther
recipient
transfer

# Summary

Token is an interesting project that has a friendly and growing community. The Smart Contract analysis reported no compiler error or critical issues. The contract Owner can access some admin functions that can not be used in a malicious way to disturb the users' transactions. There is also a limit of max 25% fees.

There are some functions that can be abused by the owner, like manipulating fees and transferring funds to the team's wallet. The maximum fee percentage that can be set is 25%. A multi-wallet signing pattern will provide security against potential hacks. Temporarily locking the contract or renouncing ownership will eliminate all the contract threats.

The contract can be converted into a honeypot and prevent users from selling if the owner abuses the admin functions.

This audit investigates security issues, business logic concerns and potential improvements.

# 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 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