BlockSAFU

ADVANCE MANUAL
SMART CONTRACT AUDIT

Project: APEGUILD

Website: hold4gold.org

PASSED

BlockSAFU Score:

95

Contract Address:

0xD5Ec1460807EEEe1a1fde79019589C0562afaD99

DISCLAMER

BlockSAFU has completed this report to provide a summary of the Smart Contract functions, and any security, dependency, or cybersecurity vulnerabilities. This is often a constrained report on our discoveries based on our investigation and understanding of the current programming versions as of this report's date. To understand the full scope of our analysis, it is vital for you to at the date of this report. To understand the full scope of our analysis, you need to review the complete report. Although we have done our best in conducting our investigation and creating this report, it is vital to note that you should not depend on this report and cannot make any claim against BlockSAFU or its Subsidiaries and Team members on the premise of what has or has not been included in the report. Please remember to conduct your independent examinations before making any investment choices. We do not provide investment advice or in any way claim to determine if the project will be successful or not.

By perusing this report or any portion of it, you concur to the terms of this disclaimer. In the unlikely situation where you do not concur to the terms, you should immediately terminate reading this report, and erase and discard any duplicates of this report downloaded and/or printed by you. This report is given for data purposes as it were and on a non-reliance premise and does not constitute speculation counsel. No one should have any right to depend on the report or its substance, and BlockSAFU and its members (including holding companies, shareholders, backups, representatives, chiefs, officers, and other agents) BlockSAFU and its subsidiaries owe no obligation of care towards you or any other person, nor does BlockSAFU make any guarantee or representation to any individual on the precision or completeness of the report.

ABOUT THE AUDITOR:

BlockSAFU (BSAFU) is an Anti-Scam Token Utility that reviews Smart Contracts and Token information to Identify Rug Pull and Honey Pot scamming activity. BlockSAFUs Development Team consists of several Smart Contract creators, Auditors Developers, and Blockchain experts. BlockSAFU provides solutions, prevents, and hunts down scammers. BSAFU is a utility token with features Audit, KYC, Token Generators, and Bounty Scammers. It will enrich the crypto ecosystem.

OVERVIEW

BlockSAFU was commissioned by APEGUILD to complete a Smart Contract audit. The objective of the Audit is to achieve the following:

- Review the Project and experience and Development team
- Ensure that the Smart Contract functions are necessary and operate as intended.
- Identify any vulnerabilities in the Smart Contract code.

DISCLAIMER: This Audit is intended to inform about token Contract Risks, the result does not imply an endorsement or provide financial advice in any way, all investments are made at your own risk. (https://blocksafu.com/)

SMART CONTRACT REVIEW

Token Name	Future Gold
Token Symbol	FutureGold
Token Decimal	18
Total Supply	2,000,000,000 FutureGold
Contract Address	0xD5Ec1460807EEEe1a1fde79019589C0562afaD99
Deployer Address	0xF8868B932B6E1A8D5Acb5615E35Bcb1D0933177e
Owner Address	0xF8868B932B6E1A8D5Acb5615E35Bcb1D0933177e
Tax Fees Buy	9%
Tax Fees Sell	9%
Gas Used for Buy	*will be updated after the DEX listing*
Gas Used for Sell	*will be updated after the DEX listing*
Contract Created	Jul-24-2022 02:29:21 AM +UTC
Initial Liquidity	*will be updated after the DEX listing*
Liquidity Status	Locked
Unlocked Date	*will be updated after the DEX listing*
Verified CA	Yes
Compiler	v0.8.4+commit.c7e474f2
Optimization	Yes with 200 runs
Sol License	MIT License
Top 5 Holders	*will be updated after the DEX listing*
Other	default evmVersion

TAX

BUY	9%	**SELL**	9%
Liquidity Fee	1%	Liquidity Fee	1%
Marketing Fee	4%	Marketing Fee	4%
Reward Fee	4%	Reward Fee	4%

TOP HOLDER

Rank	Address	Quantity	Percentage	Analytics
1	0xf8668b932b6e1a8d5acb5815e35bcb1d0933177e	1,965,000,000	98.2500%	
2	0x08a013b4c77fd5340230f5ff9d84ebe4e6dc683f	35,000,000	1.7500%	

[Download CSV Export]

Team Review

The Future Gold team has a nice website, their website is professionally built and the Smart contract is well developed, their social media is growing with over 4,250 people in their telegram group (count in audit date).

OFFICIAL WEBSITE AND SOCIAL MEDIA

Website: https://hold4gold.org/

Telegram Group: https://t.me/futuregoldann

Twitter: https://twitter.com/futuregoldann

MANUAL CODE REVIEW

🟢 Minor-risk

1 minor-risk code issue found

Could be fixed, and will not bring problems.

1. The return value of an external transfer/transferFrom return value is checked. Recommendation: use SafeERC20, or ensure that the transfer/transferFrom return value is checked

```
function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);
```

🟡 Medium-risk

0 medium-risk code issues found

Should be fixed, could bring problems.

🔴 High-Risk

0 high-risk code issues found

Must be fixed, and will bring problem.

🔴 Critical-Risk

0 critical-risk code issues found

Must be fixed, and will bring problem.

EXTRA NOTES SMART CONTRACT

1. IERC20

```
interface IERC20 {
    /**
     * @dev Returns the number of tokens in existence.
     */
    function totalSupply() external view returns (uint256);
...
    function balanceOf(address account) external view returns (uint256);
...
    function transfer(address recipient, uint256 amount) external returns (bool);
...
    function allowance(address owner, address spender) external view returns (uint256);
...
    function approve(address spender, uint256 amount) external returns (bool);
...
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);

    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to, uint256 value);
...
}
```

IERC20 Normal Base Template

2. SafeMath Contract

```
library SafeMath {
...
    function add(uint256 a, uint256 b) internal pure returns
(uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }
...
    function sub(uint256 a, uint256 b, string memory errorMessage)
internal pure returns (uint256) {
        require(b <= a, errorMessage);
        uint256 c = a - b;

        return c;
    }
    /**
     * @dev Returns the multiplication of two unsigned integers,
reverting on
     * overflow.
     *
     * Counterpart to Solidity's `*` operator.
     *
     * Requirements:
     *
     * - Multiplication cannot overflow.
     */
  ...
    function mod(
        uint256 a,
        uint256 b,
        string memory errorMessage
    ) internal pure returns (uint256) {
        unchecked {
            require(b > 0, errorMessage);
            return a % b;
        }
    }
}
```

Standard Safemath contract

3. Future Gold Contract

```
contract BABYTOKENDividendTracker is OwnableUpgradeable,
DividendPayingToken {
    using SafeMath for uint256;
    using SafeMathInt for int256;
    using IterableMapping for IterableMapping.Map;

    IterableMapping.Map private tokenHoldersMap;
    uint256 public lastProcessedIndex;

    mapping(address => bool) public excludedFromDividends;

    mapping(address => uint256) public lastClaimTimes;

    uint256 public claimWait;
    uint256 public minimumTokenBalanceForDividends;

    event ExcludeFromDividends(address indexed account);
    event ClaimWaitUpdated(uint256 indexed newValue, uint256
indexed oldValue);

    event Claim(
        address indexed account,
        uint256 amount,
        bool indexed automatic
    );

    function initialize(
        address rewardToken_,
        uint256 minimumTokenBalanceForDividends_
    ) external initializer {
        DividendPayingToken.__DividendPayingToken_init(
            rewardToken_,
            "DIVIDEND_TRACKER",
            "DIVIDEND_TRACKER"
        );
        claimWait = 3600;
        minimumTokenBalanceForDividends =
minimumTokenBalanceForDividends_;
    }

    function _transfer(
```

```
        address,
        address,
        uint256
    ) internal pure override {
        require(false, "Dividend_Tracker: No transfers allowed");
    }

    function withdrawDividend() public pure override {
        require(
            false,
            "Dividend_Tracker: withdrawDividend disabled. Use the
'claim' function on the main BABYTOKEN contract."
        );
    }

    function excludeFromDividends(address account) external
onlyOwner {
        require(!excludedFromDividends[account]);
        excludedFromDividends[account] = true;

        _setBalance(account, 0);
        tokenHoldersMap.remove(account);

        emit ExcludeFromDividends(account);
    }

    function isExcludedFromDividends(address account)
        public
        view
        returns (bool)
    {
        return excludedFromDividends[account];
    }

    function updateClaimWait(uint256 newClaimWait) external
onlyOwner {
        require(
            newClaimWait >= 3600 && newClaimWait <= 86400,
            "Dividend_Tracker: claimWait must be updated to
between 1 and 24 hours"
        );
        require(
```

```
            newClaimWait != claimWait,
            "Dividend_Tracker: Cannot update claimWait to same
value"
        );
        emit ClaimWaitUpdated(newClaimWait, claimWait);
        claimWait = newClaimWait;
    }

    function updateMinimumTokenBalanceForDividends(uint256 amount)
        external
        onlyOwner
    {
        minimumTokenBalanceForDividends = amount;
    }

    function getLastProcessedIndex() external view returns
(uint256) {
        return lastProcessedIndex;
    }

    function getNumberOfTokenHolders() external view returns
(uint256) {
        return tokenHoldersMap.keys.length;
    }

    function getAccount(address _account)
        public
        view
        returns (
            address account,
            int256 index,
            int256 iterationsUntilProcessed,
            uint256 withdrawableDividends,
            uint256 totalDividends,
            uint256 lastClaimTime,
            uint256 nextClaimTime,
            uint256 secondsUntilAutoClaimAvailable
        )
    {
        account = _account;

        index = tokenHoldersMap.getIndexOfKey(account);
```

```
        iterationsUntilProcessed = -1;

        if (index >= 0) {
            if (uint256(index) > lastProcessedIndex) {
                iterationsUntilProcessed = index.sub(
                    int256(lastProcessedIndex)
                );
            } else {
                uint256 processesUntilEndOfArray =
tokenHoldersMap.keys.length >
                    lastProcessedIndex
                    ?
tokenHoldersMap.keys.length.sub(lastProcessedIndex)
                    : 0;

                iterationsUntilProcessed = index.add(
                    int256(processesUntilEndOfArray)
                );
            }
        }

        withdrawableDividends = withdrawableDividendOf(account);
        totalDividends = accumulativeDividendOf(account);

        lastClaimTime = lastClaimTimes[account];

        nextClaimTime = lastClaimTime > 0 ?
lastClaimTime.add(claimWait) : 0;

        secondsUntilAutoClaimAvailable = nextClaimTime >
block.timestamp
            ? nextClaimTime.sub(block.timestamp)
            : 0;
    }

    function getAccountAtIndex(uint256 index)
        public
        view
        returns (
            address,
            int256,
```

```
            int256,
            uint256,
            uint256,
            uint256,
            uint256,
            uint256
        )
    {
        if (index >= tokenHoldersMap.size()) {
            return (address(0), -1, -1, 0, 0, 0, 0, 0);
        }

        address account = tokenHoldersMap.getKeyAtIndex(index);

        return getAccount(account);
    }

    function canAutoClaim(uint256 lastClaimTime) private view
returns (bool) {
        if (lastClaimTime > block.timestamp) {
            return false;
        }

        return block.timestamp.sub(lastClaimTime) >= claimWait;
    }

    function setBalance(address payable account, uint256
newBalance)
        external
        onlyOwner
    {
        if (excludedFromDividends[account]) {
            return;
        }
        if (newBalance >= minimumTokenBalanceForDividends) {
            _setBalance(account, newBalance);
            tokenHoldersMap.set(account, newBalance);
        } else {
            _setBalance(account, 0);
            tokenHoldersMap.remove(account);
        }
        processAccount(account, true);
```

```
    }

    function process(uint256 gas)
        public
        returns (
            uint256,
            uint256,
            uint256
        )
    {
        uint256 numberOfTokenHolders =
tokenHoldersMap.keys.length;

        if (numberOfTokenHolders == 0) {
            return (0, 0, lastProcessedIndex);
        }

        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);
    }

    function processAccount(address payable account, bool
automatic)
        public
        onlyOwner
        returns (bool)
    {
        uint256 amount = _withdrawDividendOfUser(account);
        if (amount > 0) {
            lastClaimTimes[account] = block.timestamp;
            emit Claim(account, amount, automatic);
            return true;
        }
        return false;
    }
}
```

4. Tax Fee contract

```
function setTokenRewardsFee(uint256 value) external onlyOwner {
        tokenRewardsFee = value;
        totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
    }

    function setLiquiditFee(uint256 value) external onlyOwner {
        liquidityFee = value;
        totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
    }

    function setMarketingFee(uint256 value) external onlyOwner {
        marketingFee = value;
        totalFees = tokenRewardsFee.add(liquidityFee).add(marketingFee);
        require(totalFees <= 25, "Total fee is over 25%");
    }
```

The owner can’t set fees over 25%

4. PinkAntiBot

```
interface IPinkAntiBot {
  function setTokenOwner(address owner) external;

  function onPreTransferCheck(
    address from,
    address to,
    uint256 amount
  ) external;
}
...
 function setEnableAntiBot(bool _enable) external onlyOwner {
        enableAntiBot = _enable;
    }
```

The owner can set antibot to enable or not.

BlockSAFU TOKEN SCANNER

https://blocksafu.com/token-scanner

Token Information

Indicator	Value
Token Name	Future Gold
Token Symbol	FutureGold
Total Supply	2,000,000,000
Already Listed On Dex	Already Listed
Dex Listed	PancakeV2
Open Source	Open Source
Price	$NaN
Volume 24H	$NaN
Liquidity	$NaN (NaN BNB)
Tx Count 24H	
Marketcap	$NaN

Security Information

Indicator	Value
Honeypot	Liquidity Not Available
Buy Fees	0%
Sell Fees	0%
Buy Gas	0 **Gwei (0.000000 BNB / $0.00)**
Sell Gas	0 **Gwei (0.000000 BNB / $0.00)**
Holder Count	3 **Holders**

Honeypot Safety

Indicator	Value
Can Take Back Ownership	Not detected
Owner Change Balance	Not detected
Blacklist	Not detected
Modify Fees	Detected
Proxy	Not detected
Whitelisted	Not detected
Anti Whale	Detected
Trading Cooldown	Not detected
Transfer Pausable	Not detected
Cannot Sell All	Not detected

Rug Pull Safety

Indicator	Value
Hidden Owner	Not detected
Creator Address	0xf8868b93...77e
Creator Balance	0 FutureGold
Creator Percent	0%
Owner Address	0xf8868b93...77e
Owner Balance	0 FutureGold
Owner Percent	0%
Lp Holder Count	0
Lp Total Supply	NaN
Mint	Not detected

Top Holder Held

Top Holder Held
54.027%

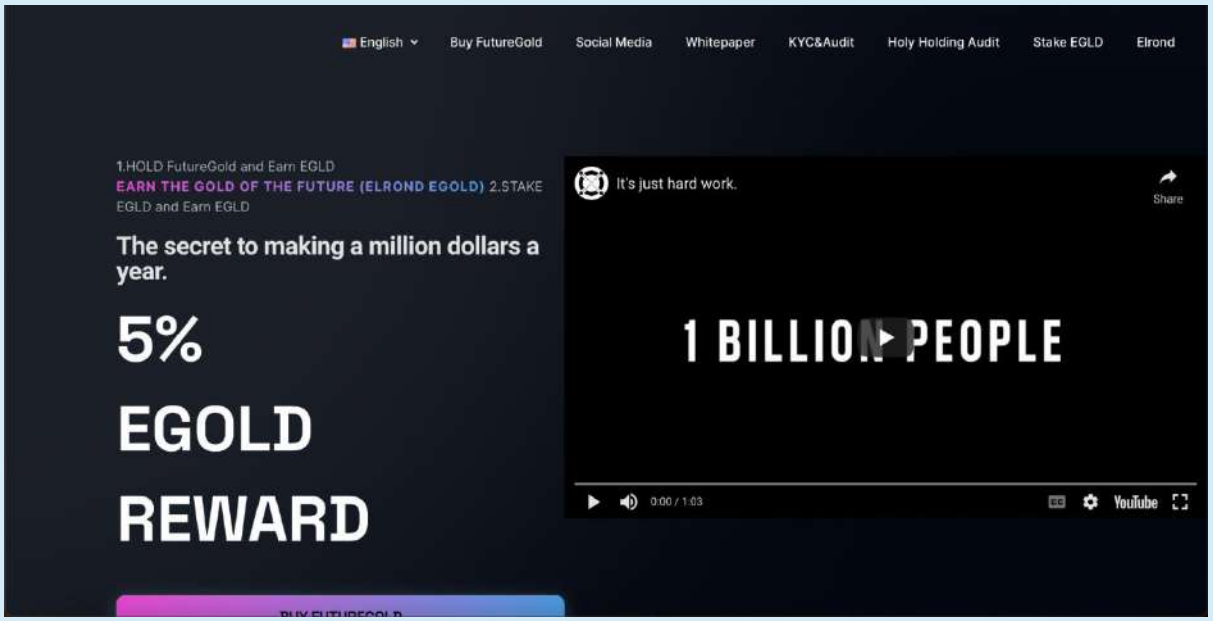
Address	Balance	Percent	Tag
0x0000...ead	1,045,546,219.2	52.277%	-
0x08a0...83f	35,000,000	1.750%	-

Liquidity Information

Note:
This is Contract Address
This is Wallet Address
Lp Locked

Address	Balance	Percent	Tag	Unlocked Date

WEBSITE REVIEW



- **Mobile Friendly**
- **Contains no code error**
- **SSL Secured (By E1 SSL)**

Web-Tech stack: Core, Swiper

Domain .finance (namecheap) - Tracked by whois

First Contentful Paint:	515ms
Fully Loaded Time	3.8s
Performance	69%
Accessibility	94%
Best Practices	67%
SEO	92%

RUG-PULL REVIEW

Based on the available information analyzed by us, we come to the following conclusions:

- Locked Liquidity (Locked by pinksale)

  (*Will be updated after DEX listing*)

- TOP 5 Holder.

  (*Will be updated after DEX listing*)

- The Team has Passed KYC by SolidProof

(https://github.com/solidproof/projects/blob/main/Future%20Gold/KYC_Certificate_Solidproof_Future_Gold.png)

HONEYPOT REVIEW

- Ability to sell.
- The owner is not able to pause the contract.
- The owner can’t set fees over 25%
- PinkAntiBot

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. The report is provided for the only contract mentioned in the report and does not include any other potential contracts deployed by the project owner.