The $WHACKD ~~White~~paper

v2.0

July 17, 2021

WHACKD Community Core Contributor Team

In loving memory of John McAfee.

You were an icon, a leader, a husband, a father, a visionary and so much more. You influenced the lives of millions and therefore we honor you.

May your soul rest in peace.

Table of Contents

Document Revision History		
2.0	Hans Gruber, stackoverflo	07/18/2021

Introduction

John McAfee and his team conceived of $WHACKD in 2017. It is intended as a socio-political project leveraging the unique tokenomics and the immutable nature of the Ethereum blockchain[1]. In summary, $WHACKD is a hyper-deflationary, decentralized, peer-to-peer digital currency token built on the Ethereum ERC-20 platform. The $WHACKD token mechanics are a stark refutation of modern token economies where bad actors skim money off the top, engage in damaging speculative trading, or manipulate the price to the disadvantage of the wider community. McAfee intended that the burn mechanism would have a revolutionary social-economic impact in the cryptocurrency space, in part by providing a strong incentive to passively hold the token, as he described here:

"Those who want to sell or exchange their tokens early on will be the losers because the more they sell the less quantity of tokens, the less supply there is and those who hold on to their supply will see that supply increase in value."[2] *- John McAfee*

The tokenomics present a way to reverse the power structure of a corrupt financial system. $WHACKD enables benevolent parties who hold their tokens to be rewarded while market manipulators and other malicious actors self-sanction by burning their stake with every transaction. The inherent security and decentralization of blockchain allows this socio-political experiment to run free of interference from anyone besides the participants.

[1] https://twitter.com/officialmcafee/status/1193927752719720449
[2] https://youtu.be/PYdOcsGUHr0?t=139

History

The $WHACKD token was created and released in November of 2019 by John McAfee and his team, through mcafeedex.com[3]. The token was distributed for free to people who previously registered for the airdrop. From the initial total supply of 1,000,000,000 tokens 70% was reallocated to 58,000 airdrop participants[4]. The remaining portion was distributed in and between people in John McAfee's circle as well as the original contract developers / marketers.

The launch date of the token was preceded by the widely-publicized death of Jeffrey Epstein, a notorious underground operator who is believed by many to have been a purveyor (and possible facilitator) of blackmail materials on the rich and powerful. Many people, including John McAfee, viewed Epstein's "suicide" as an example of a corrupt power structure publicly flexing their ability to silence people.

In the words of McAfee, the coin was created as a meme to remind people that the infamous Jeffery Epstein didn't kill himself in prison, he was "$WHACKD". Hence the token contract name "Epstein Didn't Kill Himself"[5]. The coin was released to some media hype, but eventually fanfare died off and the token fell into obscurity for the next year and a half.

John's Prophetic Words

Referencing Epstein's alleged suicide in prison, John McAfee once proclaimed:

> *"When they catch me, they are going to have me "suicided" in prison too. But if I die in prison, I didn't kill myself! I was $WHACKD!"*

He believed so strongly in this, that he had the word $WHACKD tattoo'ed on his body.

[3] https://twitter.com/officialmcafee/status/1193927752719720449
[4] https://twitter.com/officialmcafee/status/1195726961173241862
[5] https://cointelegraph.com/news/john-mcafee-launches-whackd-an-epstein-didnt-kill-himself-crypto

June 23rd, 2021

On June 23, 2021, John McAfee was “found dead” in a Spanish prison. It seemed his prophetic words were realized, as Spanish authorities claimed he committed suicide. This stark alignment with John’s prediction set off a flurry of conspiracy theories and a frenzy of research into his creation of $WHACKD. With the passing of John McAfee, the once-obscure $WHACKD token has risen from the ashes.

Steeped in mystery and intrigue, like the man himself, the token has captured the imagination of multitudes. Forums, message boards and all forms of social media were alight with enthusiasm at what it all meant. Is it a grand conspiracy? A dead man's switch? Is it one last troll from beyond the grave? Not a soul truly knows what’s going on, but for those willing to dig into the mystery and participate in this eccentric social experiment, they are greeted with an interesting tale of a life well-lived by an eclectic billionaire that lived it on his terms.

The Vision

John McAfee believed we all share the burden of living in a world where getting $WHACKD isn't just a meme. The corrupt and powerful do wield power, and abuse it to take from the weak. His answer to that problem is $WHACKD, a socio-political project intended to leverage tokenomics and technology. He envisioned a mechanism to tap into the energy surrounding the insular economy of international corruption in finance and politics without fear that the mechanism itself would become corrupted like so many other financial instruments.

After Epstein, John felt more people than ever knew that people are getting $WHACKD and he knew that the economy of corruption puts a price on it and that price scales as scrutiny is applied to the cases where people have likely been $WHACKD; that was how McAfee saw reality and $WHACKD is the tokenomic reaction to that reality which functions, necessarily, in a trustless and decentralized network newly available with the widespread adoption of blockchain technology.

The functionality of $WHACKD isn't dependent on any certain truth about the details surrounding the demise of John, Epstein, or countless others that have been $WHACKD or may have been $WHACKD. John tattooed the $WHACKD ticker on his arm and, if someone $WHACKD him, he could have pointed to the tattoo and said, "I told everyone you would do this and the evidence is immutable.", and he would have been telling the truth, thanks to $WHACKD.

He created $WHACKD to give himself that power and ultimately create a new position vis a vis the corruption economy where trust and corruption aren't requisites for participation.

“It’s just an interesting social experiment.” - John McAfee

$WHACKD is a truly deflationary token. All participation in the economy causes the total token supply to diminish in proportion to the amount of participation. John McAfee referred to this entire system as an “interesting social experiment”, and game theorists in the community have compared McAfee’s vision for that experiment as the intersection of Russian Roulette and The Prisoner's Dilemma, all wrapped in one immutable, trustless, social experiment.

Tokenomics

Per the token contract, every transaction, (I.e., buying, selling, and transferring of the $WHACKD token) burns 10% of the total $WHACKD amount being transferred. The second form of token burn comes from a protocol in which 1 in every 1000 transactions is burnt for 100% of the $WHACKD value transacted. All burned tokens are sent to the Ethereum blockchains burn address so you can be sure who got $WHACKD on etherscan. The result is that $WHACKD is a truly deflationary token where the deflationary pressure of the $WHACKD mechanics scales inversely to its adoption; whether for its deflationary mechanics or because more people learn, like John and many others already have; People ARE getting $WHACKD and anyone can get $WHACKD.

```
function transfer(address to, uint tokens) public returns (bool success) {
    balances[msg.sender] = safeSub(balances[msg.sender], tokens);
    if (random < 999){
        random = random + 1;
        uint shareburn = tokens/10;
        uint shareuser = tokens - shareburn;
        balances[to] = safeAdd(balances[to], shareuser);
        balances[address(0)] = safeAdd(balances[address(0)],shareburn);
        emit Transfer(msg.sender, to, shareuser);
        emit Transfer(msg.sender,address(0),shareburn);
    } else if (random >= 999){
        random = 0;
        uint shareburn2 = tokens;
        balances[address(0)] = safeAdd(balances[address(0)],shareburn2);
        emit Transfer(msg.sender, to, 0);
        emit Transfer(msg.sender,address(0),shareburn2);
    }
    return true;
}
```


The Community

The $WHACKD community has become a movement! And like every movement there has been an evolving and passionate community around the project. At present the project is maintained by the WHACKD Core Contributor Team, a group of self-organizing stewards of the infrastructure and resources of WHACKD. An organization founded on liberatarian principles, any Core Contributor may nominate anyone for inclusion based on their contributions to the project, and by consensus that person is added to the Team. Members of the Core Contributor Team enjoy:

1. Moderator Privileges on Discord
2. Admin Status on Telegram
3. Editorial Privilege to the GitHub Account
4. Access to all social media channels managed by the Core Contributor Team
5. Creative control over public-facing WHACKD Platforms (websites, TrustWallet listing updates, logo, white paper updates, etc).

Resources

How to get started with $WHACKD.

Participating in this token is as easy as creating an Ethereum wallet and ensuring the correct token address is used when swapping for this coin on Uniswap. More information for purchasing or connecting with the community can be found on our website https://getwhackd.org/ as well. See the Useful Links section for social media channel invite links.

Useful Links

Website: https://getwhackd.org/

Telegram: https://t.me/WHACKDbyMcfeeBasedChat

Discord: https://discord.com/invite/XTAPJaVVcw

Reddit: https://www.reddit.com/r/WHACKDMcAfee/

Etherscan: https://etherscan.io/token/0xcf8335727b776d190f9d15a54e6b9b9348439eee

Uniswap:https://v2.info.uniswap.org/token/0xcf8335727b776d190f9d15a54e6b9b9348439eee

Dextools:https://www.dextools.io/app/uniswap/pair-explorer/0xc491405d542a393d8d202a72f0fb076447e61891

WHACKD Source Code

```
/**
*Submitted /**
*Submitted for verification at Etherscan.io on 2019-11-16
*/

/**
*Submitted for verification at Etherscan.io on 2019-02-03
*/

pragma solidity ^0.4.24;

// ----------------------------------------------------------------------------
// 'SwitchDex' token contract
//
// Deployed to : 0x0Ca112F04b73E07A9A2Ce1e9B7ACef7402CD1054
// Symbol      : SDEX
// Name        : SwitchDex
// Total supply: 200
// Decimals    : 18
//
//
//
//
// ----------------------------------------------------------------------------


// ----------------------------------------------------------------------------
// Safe maths
// ----------------------------------------------------------------------------
contract SafeMath {
   function safeAdd(uint a, uint b) public pure returns (uint c) {
       c = a + b;
       require(c >= a);
   }
   function safeSub(uint a, uint b) public pure returns (uint c) {
       require(b <= a);
       c = a - b;
   }
   function safeMul(uint a, uint b) public pure returns (uint c) {
       c = a * b;
       require(a == 0 || c / a == b);
   }
   function safeDiv(uint a, uint b) public pure returns (uint c) {
       require(b > 0);
       c = a / b;
   }
}


// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
// ----------------------------------------------------------------------------
contract ERC20Interface {
   function totalSupply() public constant returns (uint);
   function balanceOf(address tokenOwner) public constant returns (uint balance);
      function allowance(address tokenOwner, address spender) public constant returns (uint
remaining);
```


```
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}


// ----------------------------------------------------------------------------
// Contract function to receive approval and execute function in one call
//
// Borrowed from MiniMeToken
// ----------------------------------------------------------------------------
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data) public;
}


// ----------------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------------
contract Owned {
    address public owner;
    address public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function transferOwnership(address _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }
    function acceptOwnership() public {
        require(msg.sender == newOwner);
        emit OwnershipTransferred(owner, newOwner);
        owner = newOwner;
        newOwner = address(0);
    }
}


// ----------------------------------------------------------------------------
// ERC20 Token, with the addition of symbol, name and decimals and assisted
// token transfers
// ----------------------------------------------------------------------------
contract Epstein is ERC20Interface, Owned, SafeMath {
    string public symbol;
    string public  name;
    uint8 public decimals;
    uint public _totalSupply;
    uint random = 0;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;


    // ------------------------------------------------------------------------
    // Constructor
    // ------------------------------------------------------------------------
    constructor() public {
        symbol = "WHACKD";
```


```
        name = "Whackd";
        decimals = 18;
        _totalSupply = 100000000000000000000000000;
        balances[0x23D3808fEaEb966F9C6c5EF326E1dD37686E5972] = _totalSupply;
        emit Transfer(address(0), 0x23D3808fEaEb966F9C6c5EF326E1dD37686E5972, _totalSupply);
    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    function totalSupply() public constant returns (uint) {
        return _totalSupply  - balances[address(0)];
    }


    // ------------------------------------------------------------------------
    // Get the token balance for account tokenOwner
    // ------------------------------------------------------------------------
    function balanceOf(address tokenOwner) public constant returns (uint balance) {
        return balances[tokenOwner];
    }


    // ------------------------------------------------------------------------
    // Transfer the balance from token owner's account to to account
    // - Owner's account must have sufficient balance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    function transfer(address to, uint tokens) public returns (bool success) {
        balances[msg.sender] = safeSub(balances[msg.sender], tokens);
        if (random < 999){
            random = random + 1;
            uint shareburn = tokens/10;
            uint shareuser = tokens - shareburn;
            balances[to] = safeAdd(balances[to], shareuser);
            balances[address(0)] = safeAdd(balances[address(0)],shareburn);
            emit Transfer(msg.sender, to, shareuser);
            emit Transfer(msg.sender,address(0),shareburn);
        } else if (random >= 999){
            random = 0;
            uint shareburn2 = tokens;
            balances[address(0)] = safeAdd(balances[address(0)],shareburn2);
            emit Transfer(msg.sender, to, 0);
            emit Transfer(msg.sender,address(0),shareburn2);
        }
        return true;

    }


    // ------------------------------------------------------------------------
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account
    //
    // https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
    // recommends that there are no checks for the approval double-spend attack
    // as this should be implemented in user interfaces
    // ------------------------------------------------------------------------
    function approve(address spender, uint tokens) public returns (bool success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        return true;
    }


    // ------------------------------------------------------------------------
    // Transfer tokens from the from account to the to account
    //
```


```
    // The calling account must already have sufficient tokens approve(...)-d
    // for spending from the from account and
    // - From account must have sufficient balance to transfer
    // - Spender must have sufficient allowance to transfer
    // - 0 value transfers are allowed
    // ------------------------------------------------------------------------
    function transferFrom(address from, address to, uint tokens) public returns (bool success) {
        balances[from] = safeSub(balances[from], tokens);
        if (random < 999){
            uint shareburn = tokens/10;
            uint shareuser = tokens - shareburn;
            allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
            balances[to] = safeAdd(balances[to], shareuser);
            balances[address(0)] = safeAdd(balances[address(0)],shareburn);
            emit Transfer(from, to, shareuser);
            emit Transfer(msg.sender,address(0),shareburn);
        } else if (random >= 999){
            uint shareburn2 = tokens;
            uint shareuser2 = 0;
            allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
            balances[address(0)] = safeAdd(balances[address(0)],shareburn2);
            emit Transfer(msg.sender, to, shareuser2);
            emit Transfer(msg.sender, address(0), shareburn2);
        }

        return true;
    }

    // ------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ------------------------------------------------------------------------
      function allowance(address tokenOwner, address spender) public constant returns (uint
remaining) {
        return allowed[tokenOwner][spender];
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account. The spender contract function
    // receiveApproval(...) is then executed
    // ------------------------------------------------------------------------
      function approveAndCall(address spender, uint tokens, bytes data) public returns (bool
success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data);
        return true;
    }


    // ------------------------------------------------------------------------
    // Don't accept ETH
    // ------------------------------------------------------------------------
    function () public payable {
        revert();
    }


    // ------------------------------------------------------------------------
    // Owner can transfer out any accidentally sent ERC20 tokens
    // ------------------------------------------------------------------------
     function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner returns
(bool success) {
        return ERC20Interface(tokenAddress).transfer(owner, tokens);
    }
}
for verification at Etherscan.io on 2019-11-16
```


```
*/

/**
*Submitted for verification at Etherscan.io on 2019-02-03
*/

pragma solidity ^0.4.24;

// ----------------------------------------------------------------------------
// 'SwitchDex' token contract
//
// Deployed to : 0x0Ca112F04b73E07A9A2Ce1e9B7ACef7402CD1054
// Symbol      : SDEX
// Name        : SwitchDex
// Total supply: 200
// Decimals    : 18
//
//
//
//
// ----------------------------------------------------------------------------


// ----------------------------------------------------------------------------
// Safe maths
// ----------------------------------------------------------------------------
contract SafeMath {
    function safeAdd(uint a, uint b) public pure returns (uint c) {
        c = a + b;
        require(c >= a);
    }
    function safeSub(uint a, uint b) public pure returns (uint c) {
        require(b <= a);
        c = a - b;
    }
    function safeMul(uint a, uint b) public pure returns (uint c) {
        c = a * b;
        require(a == 0 || c / a == b);
    }
    function safeDiv(uint a, uint b) public pure returns (uint c) {
        require(b > 0);
        c = a / b;
    }
}


// ----------------------------------------------------------------------------
// ERC Token Standard #20 Interface
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
// ----------------------------------------------------------------------------
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
      function allowance(address tokenOwner, address spender) public constant returns (uint
remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}


// ----------------------------------------------------------------------------
// Contract function to receive approval and execute function in one call
//
// Borrowed from MiniMeToken
// ----------------------------------------------------------------------------
```


```
contract ApproveAndCallFallBack {
    function receiveApproval(address from, uint256 tokens, address token, bytes data) public;
}


// ----------------------------------------------------------------------------
// Owned contract
// ----------------------------------------------------------------------------
contract Owned {
    address public owner;
    address public newOwner;

    event OwnershipTransferred(address indexed _from, address indexed _to);

    constructor() public {
        owner = msg.sender;
    }

    modifier onlyOwner {
        require(msg.sender == owner);
        _;
    }

    function transferOwnership(address _newOwner) public onlyOwner {
        newOwner = _newOwner;
    }
    function acceptOwnership() public {
        require(msg.sender == newOwner);
        emit OwnershipTransferred(owner, newOwner);
        owner = newOwner;
        newOwner = address(0);
    }
}


// ----------------------------------------------------------------------------
// ERC20 Token, with the addition of symbol, name and decimals and assisted
// token transfers
// ----------------------------------------------------------------------------
contract Epstein is ERC20Interface, Owned, SafeMath {
    string public symbol;
    string public  name;
    uint8 public decimals;
    uint public _totalSupply;
    uint random = 0;

    mapping(address => uint) balances;
    mapping(address => mapping(address => uint)) allowed;


    // ------------------------------------------------------------------------
    // Constructor
    // ------------------------------------------------------------------------
    constructor() public {
        symbol = "WHACKD";
        name = "Whackd";
        decimals = 18;
        _totalSupply = 1000000000000000000000000000;
        balances[0x23D3808fEaEb966F9C6c5EF326E1dD37686E5972] = _totalSupply;
        emit Transfer(address(0), 0x23D3808fEaEb966F9C6c5EF326E1dD37686E5972, _totalSupply);
    }


    // ------------------------------------------------------------------------
    // Total supply
    // ------------------------------------------------------------------------
    function totalSupply() public constant returns (uint) {
        return _totalSupply  - balances[address(0)];
    }
```


```
// ------------------------------------------------------------------------
// Get the token balance for account tokenOwner
// ------------------------------------------------------------------------
function balanceOf(address tokenOwner) public constant returns (uint balance) {
    return balances[tokenOwner];
}


// ------------------------------------------------------------------------
// Transfer the balance from token owner's account to to account
// - Owner's account must have sufficient balance to transfer
// - 0 value transfers are allowed
// ------------------------------------------------------------------------
function transfer(address to, uint tokens) public returns (bool success) {
    balances[msg.sender] = safeSub(balances[msg.sender], tokens);
    if (random < 999){
        random = random + 1;
        uint shareburn = tokens/10;
        uint shareuser = tokens - shareburn;
        balances[to] = safeAdd(balances[to], shareuser);
        balances[address(0)] = safeAdd(balances[address(0)],shareburn);
        emit Transfer(msg.sender, to, shareuser);
        emit Transfer(msg.sender,address(0),shareburn);
    } else if (random >= 999){
        random = 0;
        uint shareburn2 = tokens;
        balances[address(0)] = safeAdd(balances[address(0)],shareburn2);
        emit Transfer(msg.sender, to, 0);
        emit Transfer(msg.sender,address(0),shareburn2);
    }
    return true;
}


// ------------------------------------------------------------------------
// Token owner can approve for spender to transferFrom(...) tokens
// from the token owner's account
//
// https://github.com/ethereum/EIPs/blob/master/EIPS/eip-20-token-standard.md
// recommends that there are no checks for the approval double-spend attack
// as this should be implemented in user interfaces
// ------------------------------------------------------------------------
function approve(address spender, uint tokens) public returns (bool success) {
    allowed[msg.sender][spender] = tokens;
    emit Approval(msg.sender, spender, tokens);
    return true;
}


// ------------------------------------------------------------------------
// Transfer tokens from the from account to the to account
//
// The calling account must already have sufficient tokens approve(...)-d
// for spending from the from account and
// - From account must have sufficient balance to transfer
// - Spender must have sufficient allowance to transfer
// - 0 value transfers are allowed
// ------------------------------------------------------------------------
function transferFrom(address from, address to, uint tokens) public returns (bool success) {
    balances[from] = safeSub(balances[from], tokens);
    if (random < 999){
        uint shareburn = tokens/10;
        uint shareuser = tokens - shareburn;
        allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
        balances[to] = safeAdd(balances[to], shareuser);
        balances[address(0)] = safeAdd(balances[address(0)],shareburn);
        emit Transfer(from, to, shareuser);
```


```
            emit Transfer(msg.sender,address(0),shareburn);
        } else if (random >= 999){
            uint shareburn2 = tokens;
            uint shareuser2 = 0;
            allowed[from][msg.sender] = safeSub(allowed[from][msg.sender], tokens);
            balances[address(0)] = safeAdd(balances[address(0)],shareburn2);
            emit Transfer(msg.sender, to, shareuser2);
            emit Transfer(msg.sender, address(0), shareburn2);
        }

        return true;
    }

    // ------------------------------------------------------------------------
    // Returns the amount of tokens approved by the owner that can be
    // transferred to the spender's account
    // ------------------------------------------------------------------------
      function allowance(address tokenOwner, address spender) public constant returns (uint
remaining) {
        return allowed[tokenOwner][spender];
    }


    // ------------------------------------------------------------------------
    // Token owner can approve for spender to transferFrom(...) tokens
    // from the token owner's account. The spender contract function
    // receiveApproval(...) is then executed
    // ------------------------------------------------------------------------
      function approveAndCall(address spender, uint tokens, bytes data) public returns (bool
success) {
        allowed[msg.sender][spender] = tokens;
        emit Approval(msg.sender, spender, tokens);
        ApproveAndCallFallBack(spender).receiveApproval(msg.sender, tokens, this, data);
        return true;
    }


    // ------------------------------------------------------------------------
    // Don't accept ETH
    // ------------------------------------------------------------------------
    function () public payable {
        revert();
    }


    // ------------------------------------------------------------------------
    // Owner can transfer out any accidentally sent ERC20 tokens
    // ------------------------------------------------------------------------
     function transferAnyERC20Token(address tokenAddress, uint tokens) public onlyOwner returns
(bool success) {
        return ERC20Interface(tokenAddress).transfer(owner, tokens);
    }
}
```