SMART CONTRACT AUDIT

BIT_CONTROL
SECURE BLOCKCHAIN

@BIT_CONTROL

Summary

Auditing Firm	Bit Control
Architecture	Bit Control Auditing Standard
Smart Contract Audit Approved By	Nuno | Blockchain Specialist at Bit Control
Project Overview Approved By	Ricardo | Marketing Specialist at Bit Control
Platform	Solidity
Mandatory Audit Check	Static, Software & Manual Analysis
Consultation Request Date	December 18, 2022
Report Date	December 18, 2022

Audit Summary

Bit Control team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- ★ FTMGrowHouse smart contract source code has **LOW RISK SEVERITY**.
- ★ FTMGrowHouse has **PASSED** the smart contract audit.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅ Verify the authenticity of this report on Bit Control Website: https://www.bit-control.com/

Table Of Contents

Project Overview

Bit Control was consulted by FTMGrowHouse to conduct the smart contract security audit of their solidity source code.

Project	FTMGrowHouse
Blockchain	Fantom Opera Mainnet
Language	Solidity
Contracts	0xDF330DC8F709B491063E4c316b5Ae3b3859187Ea

Public logo:

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://ftmscan.com/address/0xdf330dc8f709b491063e4c316b5ae3b3859187ea#code

Contract Name: FTMGrowHouse

Compiler Version: v0.8.16

Optimization Enabled: Yes with 200 runs

SHA-1 Hash

Solidity source code is audited at hash

#j8sn29mwk5gk3xiam3ca8c76a9bd11283d33371h786

Audit Scope & Methodology

The scope of this report is to audit the smart contract source code of FTMGrowHouse, Bit Control has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Smart Contract Vulnerabilities

- ☐ Re-entrancy
- ☐ Unhandled Exceptions
- ☐ Transaction Order Dependency
- ☐ Integer Overflow
- ☐ Unrestricted Action
- ☐ Incorrect Inheritance Order
- ☐ Typographical Errors
- ☐ Requirement Violation

Source Code Review

- ☐ Ownership Takeover
- ☐ Gas Limit and Loops
- ☐ Deployment Consistency
- ☐ Repository Consistency
- ☐ Data Consistency
- ☐ Token Supply Manipulation

Functional Assessment

- ☐ Access Control and Authorization
- ☐ Operations Trail and Event Generation
- ☐ Assets Manipulation
- ☐ Liquidity Access

Bit Control Audit Standard

The aim of Bit Control standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
   - Review of the specifications, sources, and instructions provided to Bit Control to make sure we understand the size, scope, and functionality of the smart contract.
   - Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:
   - Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   - Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts

Automated 3P frameworks used to assess the smart contract vulnerabilities

- Slither
- Consensys MythX
- Consensus Surya
- Open Zeppelin Code Analyzer
- Solidity Code Complier

Bit Control's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract: Vulnerable:

A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

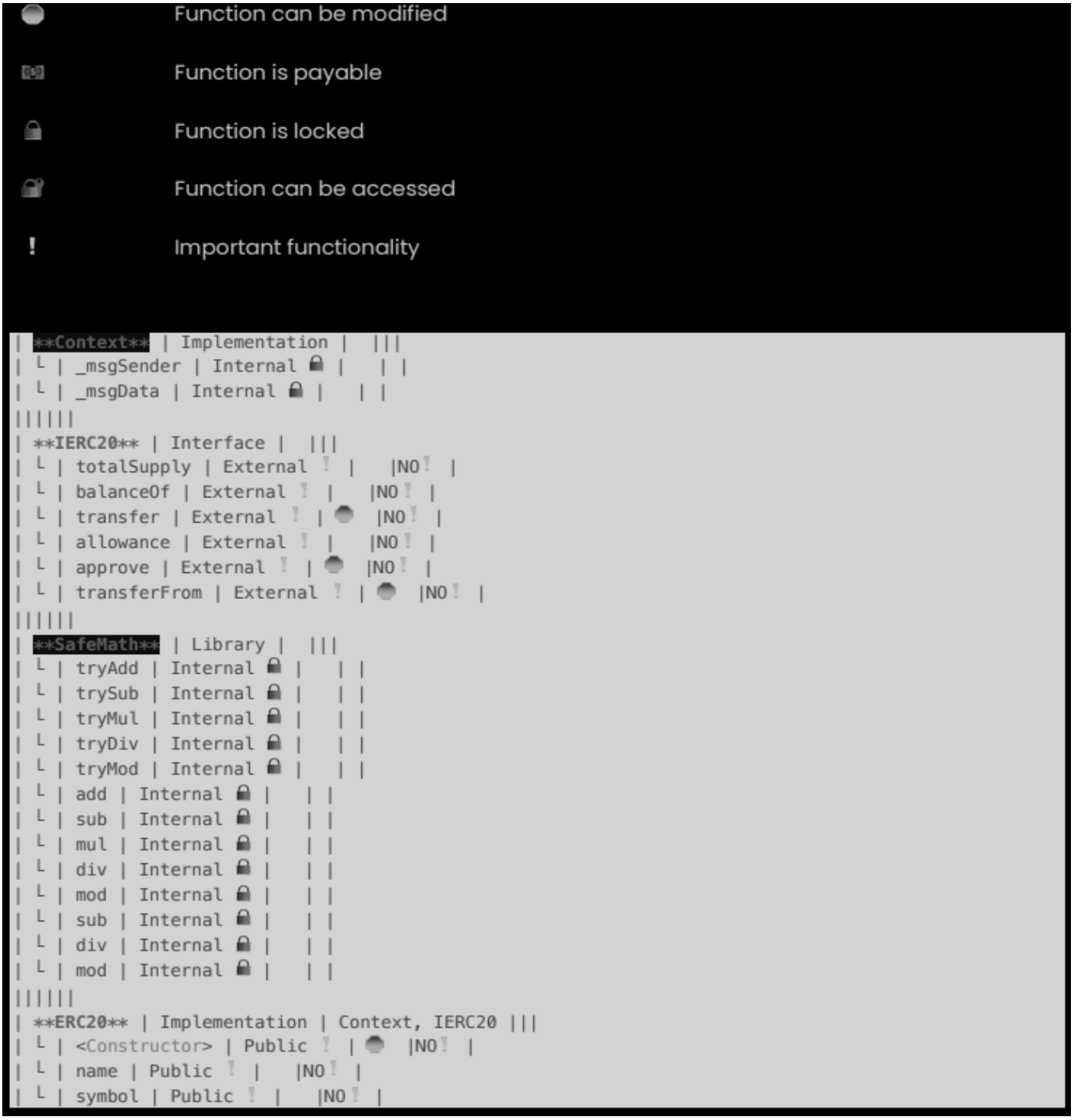
Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level of vulnerability could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity
! Medium	This level of vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level of vulnerability can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Smart Contract Risk Assessment

Static Analysis



```
| └ | decimals | Public ❗ |  |NO❗ |
| └ | totalSupply | Public ❗ |  |NO❗ |
| └ | balanceOf | Public ❗ |  |NO❗ |
| └ | transfer | Public ❗ | 🛑 |NO❗ |
| └ | allowance | Public ❗ |  |NO❗ |
| └ | approve | Public ❗ | 🛑 |NO❗ |
| └ | transferFrom | Public ❗ | 🛑 |NO❗ |
| └ | increaseAllowance | Public ❗ | 🛑 |NO❗ |
| └ | decreaseAllowance | Public ❗ | 🛑 |NO❗ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | _mint | Internal 🔒 | 🛑 | |
| └ | _burn | Internal 🔒 | 🛑 | |
| └ | _approve | Internal 🔒 | 🛑 | |
| └ | _setupDecimals | Internal 🔒 | 🛑 | |
| └ | _beforeTokenTransfer | Internal 🔒 | 🛑 | |
||||||
| **Ownable** | Implementation | Context |||
| └ | <Constructor> | Public ❗ | 🛑 |NO❗ |
| └ | owner | Public ❗ |  |NO❗ |
| └ | renounceOwnership | Public ❗ | 🛑 | onlyOwner |
| └ | transferOwnership | Public ❗ | 🛑 | onlyOwner |
||||||
| **IPinkAntiBot** | Interface |  |||
| └ | setTokenOwner | External ❗ | 🛑 |NO❗ |
| └ | onPreTransferCheck | External ❗ | 🛑 |NO❗ |
||||||
| **IAntiBotBabyToken** | Interface |  |||
| └ | initialize | External ❗ | 🛑 |NO❗ |
||||||
| **DividendPayingTokenInterface** | Interface |  |||
| └ | dividendOf | External ❗ |  |NO❗ |
| └ | withdrawDividend | External ❗ | 🛑 |NO❗ |
||||||
| **DividendPayingTokenOptionalInterface** | Interface |  |||
| └ | withdrawableDividendOf | External ❗ |  |NO❗ |
| └ | withdrawnDividendOf | External ❗ |  |NO❗ |
| └ | accumulativeDividendOf | External ❗ |  |NO❗ |
||||||
| **DividendPayingToken** | Implementation | ERC20Upgradeable, OwnableUpgradeable,
DividendPayingTokenInterface, DividendPayingTokenOptionalInterface |||
| └ | __DividendPayingToken_init | Internal 🔒 | 🛑 | initializer |
| └ | distributeCAKEDividends | Public ❗ | 🛑 | onlyOwner |
| └ | withdrawDividend | Public ❗ | 🛑 |NO❗ |
| └ | _withdrawDividendOfUser | Internal 🔒 | 🛑 | |
| └ | dividendOf | Public ❗ |  |NO❗ |
| └ | withdrawableDividendOf | Public ❗ |  |NO❗ |
| └ | withdrawnDividendOf | Public ❗ |  |NO❗ |
| └ | accumulativeDividendOf | Public ❗ |  |NO❗ |
| └ | _transfer | Internal 🔒 | 🛑 | |
| └ | _mint | Internal 🔒 | 🛑 | |
```

└	_burn	Internal 🔒	🛑		
└	_setBalance	Internal 🔒	🛑		

AntiBotBABYTOKEN	Implementation	ERC20Upgradeable, OwnableUpgradeable, IAntiBotBabyToken
└	<Constructor>	Public ❗	🛑	NO❗
└	initialize	External ❗	🛑	initializer
└	setEnableAntiBot	External ❗	🛑	onlyOwner
└	<Receive Ether>	External ❗	💸	NO❗
└	setSwapTokensAtAmount	External ❗	🛑	onlyOwner
└	updateDividendTracker	Public ❗	🛑	onlyOwner
└	updateUniswapV2Router	Public ❗	🛑	onlyOwner
└	excludeFromFees	Public ❗	🛑	onlyOwner
└	excludeMultipleAccountsFromFees	Public ❗	🛑	onlyOwner
└	setMarketingWallet	External ❗	🛑	onlyOwner
└	setTokenRewardsFee	External ❗	🛑	onlyOwner
└	setLiquiditFee	External ❗	🛑	onlyOwner
└	setMarketingFee	External ❗	🛑	onlyOwner
└	setAutomatedMarketMakerPair	Public ❗	🛑	onlyOwner
└	blacklistAddress	External ❗	🛑	onlyOwner
└	_setAutomatedMarketMakerPair	Private 🔐	🛑	
└	updateGasForProcessing	Public ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getClaimWait	External ❗		NO❗
└	getTotalDividendsDistributed	External ❗		NO❗
└	isExcludedFromFees	Public ❗		NO❗
└	withdrawableDividendOf	Public ❗		NO❗
└	dividendTokenBalanceOf	Public ❗		NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	getAccountDividendsInfo	External ❗		NO❗
└	getAccountDividendsInfoAtIndex	External ❗		NO❗
└	processDividendTracker	External ❗	🛑	NO❗
└	claim	External ❗	🛑	NO❗
└	getLastProcessedIndex	External ❗		NO❗
└	getNumberOfDividendTokenHolders	External ❗		NO❗
└	_transfer	Internal 🔒	🛑		
└	swapAndSendToFee	Private 🔐	🛑		
└	swapAndLiquify	Private 🔐	🛑		
└	swapTokensForEth	Private 🔐	🛑		
└	swapTokensForCake	Private 🔐	🛑		
└	addLiquidity	Private 🔐	🛑		
└	swapAndSendDividends	Private 🔐	🛑		

BABYTOKENDividendTracker	Implementation	OwnableUpgradeable, DividendPayingToken
└	initialize	External ❗	🛑	initializer
└	_transfer	Internal 🔒		
└	withdrawDividend	Public ❗		NO❗
└	excludeFromDividends	External ❗	🛑	onlyOwner
└	updateClaimWait	External ❗	🛑	onlyOwner
└	getLastProcessedIndex	External ❗		NO❗

Software Analysis

Function Signatures

```
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
18160ddd  =>  totalSupply()
313ce567  =>  decimals()
95d89b41  =>  symbol()
06fdde03  =>  name()
893d20e8  =>  getOwner()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
b6a5d7de  =>  authorize(address)
f0b37c04  =>  unauthorize(address)
2f54bf6e  =>  isOwner(address)
fe9fbb80  =>  isAuthorized(address)
f2fde38b  =>  transferOwnership(address)
c9c65396  =>  createPair(address,address)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
e8e33700  =>  addLiquidity(address,address,uint256,uint256,uint256,uint256,address,uint256)
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
5c11d795  =>
swapExactTokensForTokensSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
b6f9de95  =>  swapExactETHForTokensSupportingFeeOnTransferTokens(uint256,address[],address,uint256)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
2d48e896  =>  setDistributionCriteria(uint256,uint256)
14b6ca96  =>  setShare(address,uint256)
d0e30db0  =>  deposit()
ffb2c479  =>  process(uint256)
8c21cd52  =>  shouldDistribute(address)
5319504a  =>  distributeDividend(address)
f0fc6bca  =>  claimDividend()
28fd3198  =>  getUnpaidEarnings(address)
e68af3ac  =>  getCumulativeDividends(uint256)
db29fe12  =>  addShareholder(address)
9babdad6  =>  removeShareholder(address)
571ac8b0  =>  approveMax(address)
82bf293c  =>  setMaxWalletPercent(uint256)
cb712535  =>  _transferFrom(address,address,uint256)
```


SWC Attacks

SWC ID	Description	Verdict
SWC-101	Integer Overflow and Underflow	Passed
SWC-102	Outdated Compiler Version	! Low
SWC-103	Floating Pragma	Passed
SWC-104	Unchecked Call Return Value	Passed
SWC-105	Unprotected Ether Withdrawal	Passed
SWC-106	Unprotected SELFDESTRUCT Instruction	Passed
SWC-107	Re-entrancy	Passed
SWC-108	State Variable Default Visibility	Passed
SWC-109	Uninitialized Storage Pointer	Passed
SWC-110	Assert Violation	Passed
SWC-111	Use of Deprecated Solidity Functions	Passed
SWC-112	Delegate Call to Untrusted Callee	Passed
SWC-113	DoS with Failed Call	Passed
SWC-114	Transaction Order Dependence	Passed
SWC-115	Authorization through tx.origin	Passed
SWC-116	Block values as a proxy for time	Passed
SWC-117	Signature Malleability	Passed
SWC-118	Incorrect Constructor Name	Passed

Risk Status

Risk severity	Meaning
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	None low severity issues identified
Verified	7 functions and instances verified and checked
Safety Score	97 out of 100

Report Summary

Bit Control team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

FTMGrowHouse smart contract source code has **LOW RISK SEVERITY.**
FTMGrowHouse has **PASSED** the smart contract audit.

Note for stakeholders:

Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.
Make sure that the project team's KYC/identity is verified by an independent firm, e.g., Bit Control.
Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.
Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.
Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

Legal Advisory

Important Disclaimer

Bit Control provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code, and to provide a basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without Bit Control prior written consent.

Bit Control provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an adequate assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, Bit Control does not guarantee the explicit security of the audited smart contract.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

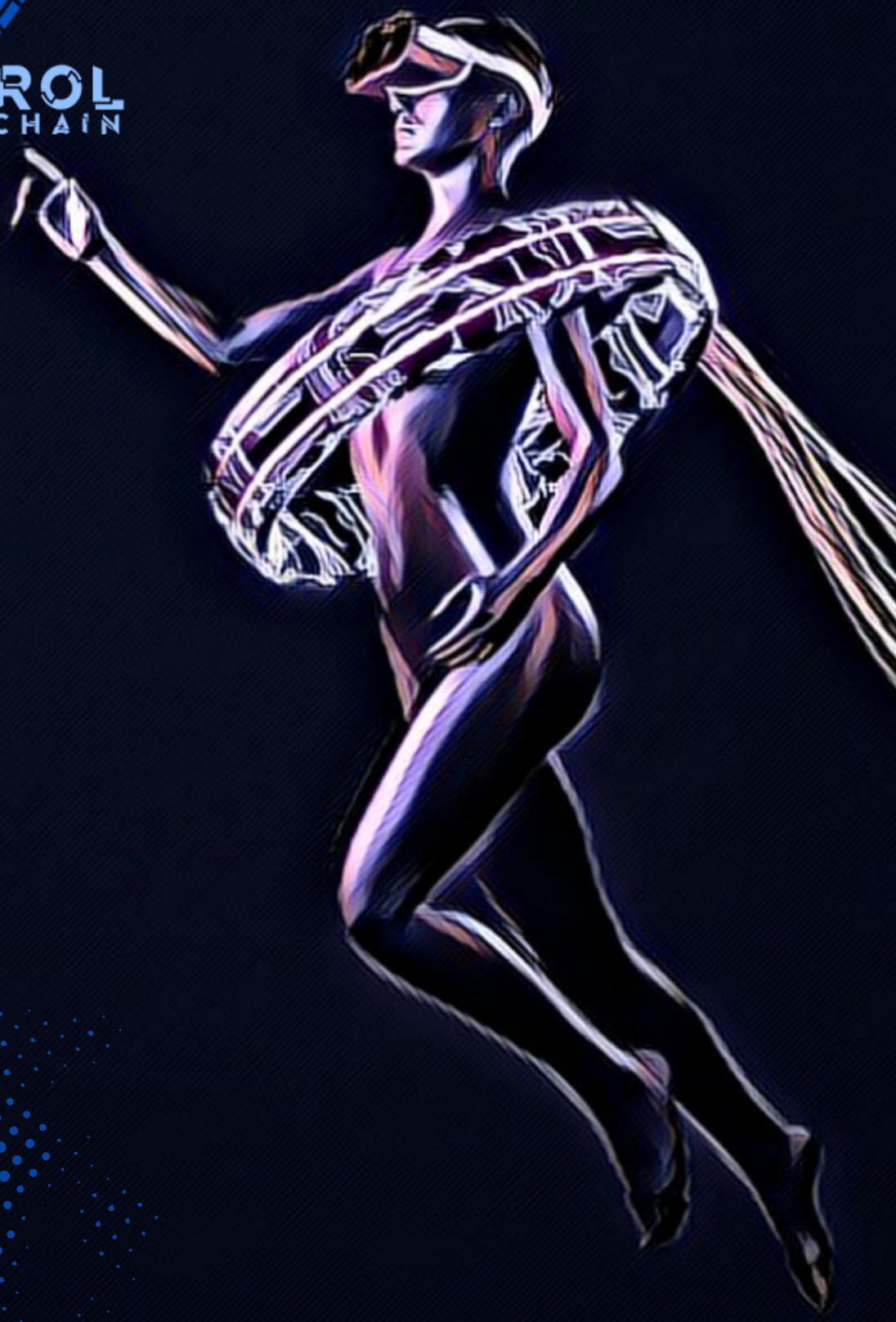
This report should not be considered as an endorsement or disapproval of any project or team. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

About Bit Control

Bit Control provides intelligent blockchain solutions. Bit Control is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. Bit Control's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

Bit Control is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 3+ core team members, and 6+ casual contributors. Bit Control provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.

BIT_CONTROL
SECURE BLOCKCHAIN



@BIT_CONTROL