DESSERT FINANCE

Bitmon Paradise (BMP)

BEP-20 Audit

Performed at block **16300577**

PERFORMED BY DESSERT FINANCE
FOR CONTRACT ADDRESS: 0xD54A91099e859c269c39d64f30F11EDE23aAF8Dd,
0X84C507EF1B6B4D2C1DE9FF2737EE1609EA01E444,
0X1DE1A1205E8EAF08FDE233ECA5B5F41CBBC14925,
0X1DE1A1205E8EAF08FDE233ECA5B5F41CBBC14925,
Vesting x3

INITIAL DISCLAIMER

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

Dessert Finance 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.

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

Table of Contents

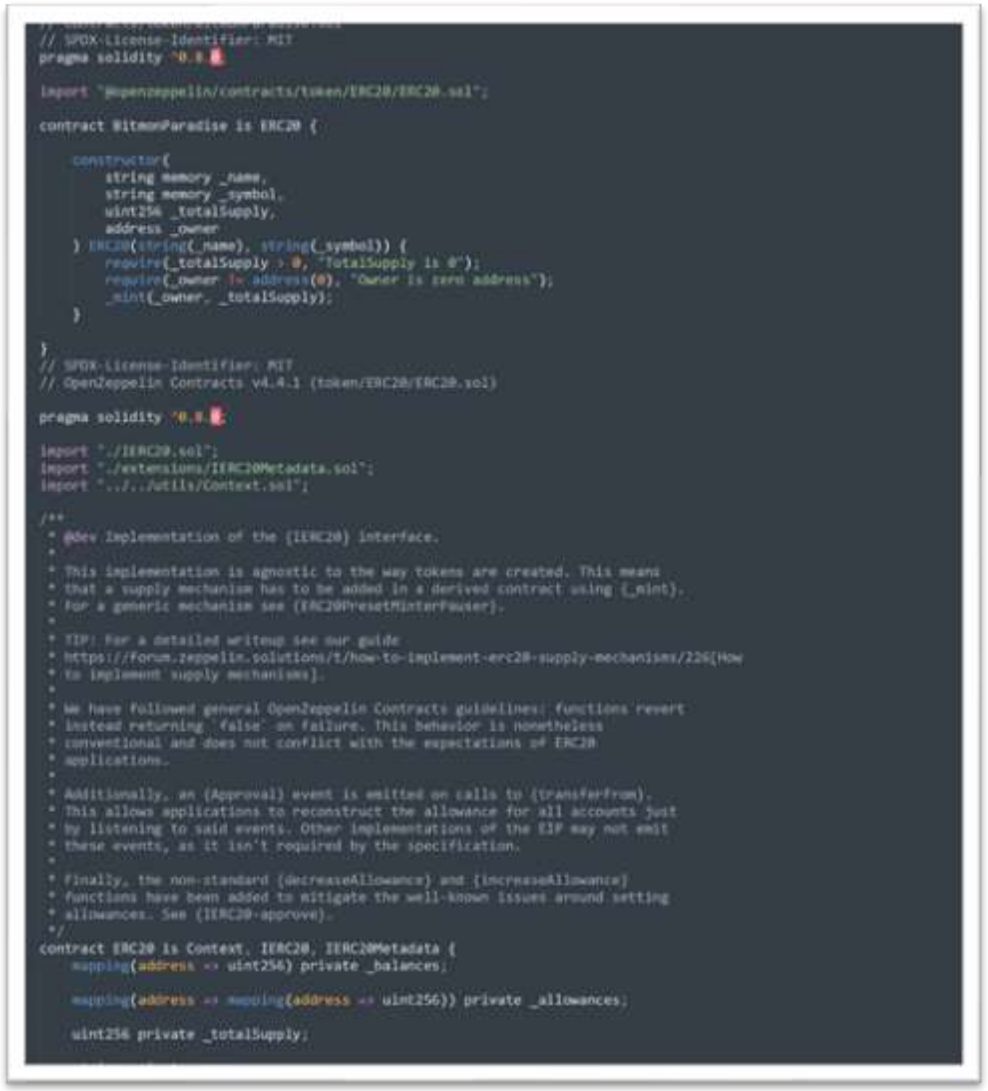
Contract Code Audit – Token Overview

Contract Name	BitmonParadise
Contract Address	0XD54A91099E859C269C39D64F30F11EDE23AAF8DD
Compiler Version	v0.8.3+commit.8d00100c
Total Supply	320,000,000 BMP
Contract Deployer Address	0xdc6Ed9A7a2AFa95A2CB920FD50a14A75248E9561
Decimals	18

DESSERT FINANCE

BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on Bitmon Paradise (BMP)



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract BitmonParadise is ERC20 {

    constructor(
        string memory _name,
        string memory _symbol,
        uint256 _totalSupply,
        address _owner
    ) ERC20(string(_name), string(_symbol)) {
        require(_totalSupply > 0, "TotalSupply is 0");
        require(_owner != address(0), "Owner is zero address");
        _mint(_owner, _totalSupply);
    }

}
// SPDX-License-Identifier: MIT
// OpenZeppelin Contracts v4.4.1 (token/ERC20/ERC20.sol)

pragma solidity ^0.8.0;

import "./IERC20.sol";
import "./extensions/IERC20Metadata.sol";
import "../../utils/Context.sol";

/**
 * @dev Implementation of the {IERC20} interface.
 *
 * This implementation is agnostic to the way tokens are created. This means
 * that a supply mechanism has to be added in a derived contract using {_mint}.
 * For a generic mechanism see {ERC20PresetMinterPauser}.
 *
 * TIP: For a detailed writeup see our guide
 * https://forum.zeppelin.solutions/t/how-to-implement-erc20-supply-mechanisms/226[How
 * to implement supply mechanisms].
 *
 * We have followed general OpenZeppelin Contracts guidelines: functions revert
 * instead returning `false` on failure. This behavior is nonetheless
 * conventional and does not conflict with the expectations of ERC20
 * applications.
 *
 * Additionally, an {Approval} event is emitted on calls to {transferFrom}.
 * This allows applications to reconstruct the allowance for all accounts just
 * by listening to said events. Other implementations of the EIP may not emit
 * these events, as it isn't required by the specification.
 *
 * Finally, the non-standard {decreaseAllowance} and {increaseAllowance}
 * functions have been added to mitigate the well-known issues around setting
 * allowances. See {IERC20-approve}.
 */
contract ERC20 is Context, IERC20, IERC20Metadata {
    mapping(address => uint256) private _balances;

    mapping(address => mapping(address => uint256)) private _allowances;

    uint256 private _totalSupply;
```

Contract Address
0xD54A91099e859c269c39d64f30F11EDE23aAF8Dd

TokenTracker
Bitmon Paradise (BMP)

Contract Creator
0xdc6Ed9A7a2AFa95A2CB920FD50a14A75248E9561

Source Code
Contract Source Code Verified

Contract Name
BitmonParadise

Other Settings
default evmVersion

Compiler Version
v0.8.3+commit.8d00100c

Optimization Enabled
Yes with 200 runs

Code is truncated to fit the constraints of this document.
The code in its entirety can be viewed here.

The contract code is **verified** on BSCScan.

BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

The contract code is **verified** on BSCScan.

The vulnerabilities listed above were not found in the token's Smart Contract.

Liquidity & Vesting – Locked / Unlocked

Vesting information has been found.

This page will contain links to locked liquidity for the project if we are able to locate that information.

Vesting Information

Three vesting contracts were found, addresses will be shown on the following pages. Token vesting has been found to be quarterly over a 2-year period.

Vesting Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

0xB719DC41b00077194f899589f5651b23AB9D55fA

The vulnerabilities listed above were not found in the token's Smart Contract.

Vesting Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

0x077F45aa2F0EEA52EF631686689D93fb33C9F7C6

The vulnerabilities listed above were not found in the token's Smart Contract.

Vesting Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

0xBA59B39fE39AE71DEB964B320F1FCC6e186F3152

The vulnerabilities listed above were not found in the token's Smart Contract.

Contract Code Audit – Mint Functions

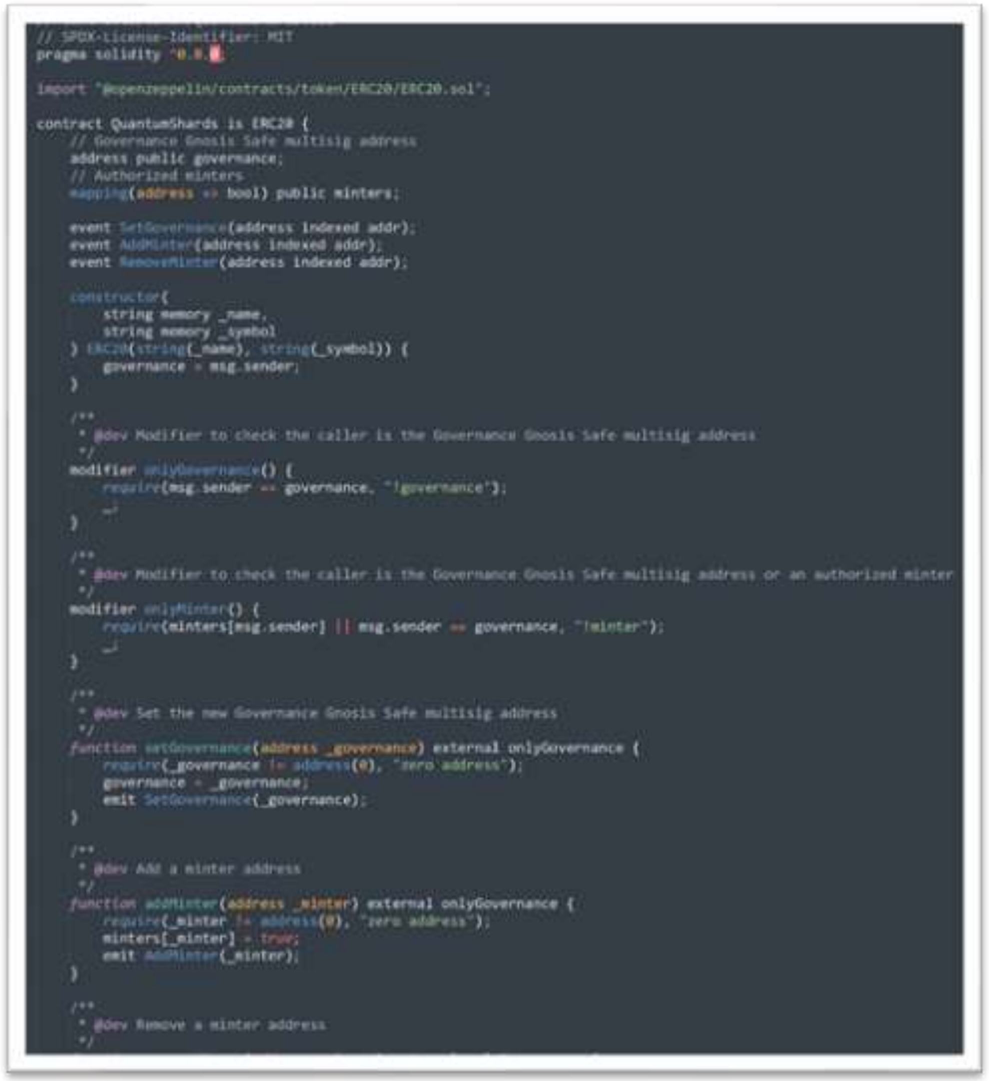
This Contract Cannot Mint New BMP Tokens.

We do understand that sometimes mint functions are essential to the functionality of the project.

A mint function was not found in the contract code.

BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on Quantum Shards (QTS)



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

contract QuantumShards is ERC20 {
    // Governance Gnosis Safe multisig address
    address public governance;
    // Authorized minters
    mapping(address => bool) public minters;

    event SetGovernance(address indexed addr);
    event AddMinter(address indexed addr);
    event RemoveMinter(address indexed addr);

    constructor(
        string memory _name,
        string memory _symbol
    ) ERC20(string(_name), string(_symbol)) {
        governance = msg.sender;
    }

    /**
     * @dev Modifier to check the caller is the Governance Gnosis Safe multisig address
     */
    modifier onlyGovernance() {
        require(msg.sender == governance, "!governance");
        _;
    }

    /**
     * @dev Modifier to check the caller is the Governance Gnosis Safe multisig address or an authorized minter
     */
    modifier onlyMinter() {
        require(minters[msg.sender] || msg.sender == governance, "!minter");
        _;
    }

    /**
     * @dev Set the new Governance Gnosis Safe multisig address
     */
    function setGovernance(address _governance) external onlyGovernance {
        require(_governance != address(0), "zero address");
        governance = _governance;
        emit SetGovernance(_governance);
    }

    /**
     * @dev Add a minter address
     */
    function addMinter(address _minter) external onlyGovernance {
        require(_minter != address(0), "zero address");
        minters[_minter] = true;
        emit AddMinter(_minter);
    }

    /**
     * @dev Remove a minter address
     */
```

Contract Address
0x84c507Ef1B6b4D2c1dE9fF2737ee1609eA01e444

TokenTracker
Quantum Shards (QTS)

Contract Creator
0xdc6Ed9A7a2AFa95A2CB920FD50a14A75248E9561

Source Code
Contract Source Code Verified

Contract Name
QuantumShards

Other Settings
default evmVersion

Compiler Version
v0.8.3+commit.8d00100c

Optimization Enabled
Yes with 200 runs

Code is truncated to fit the constraints of this document.
The code in its entirety can be viewed here.

The contract code is **verified** on BSCScan.

BEP-20 Contract Code Audit – Vulnerabilities Checked

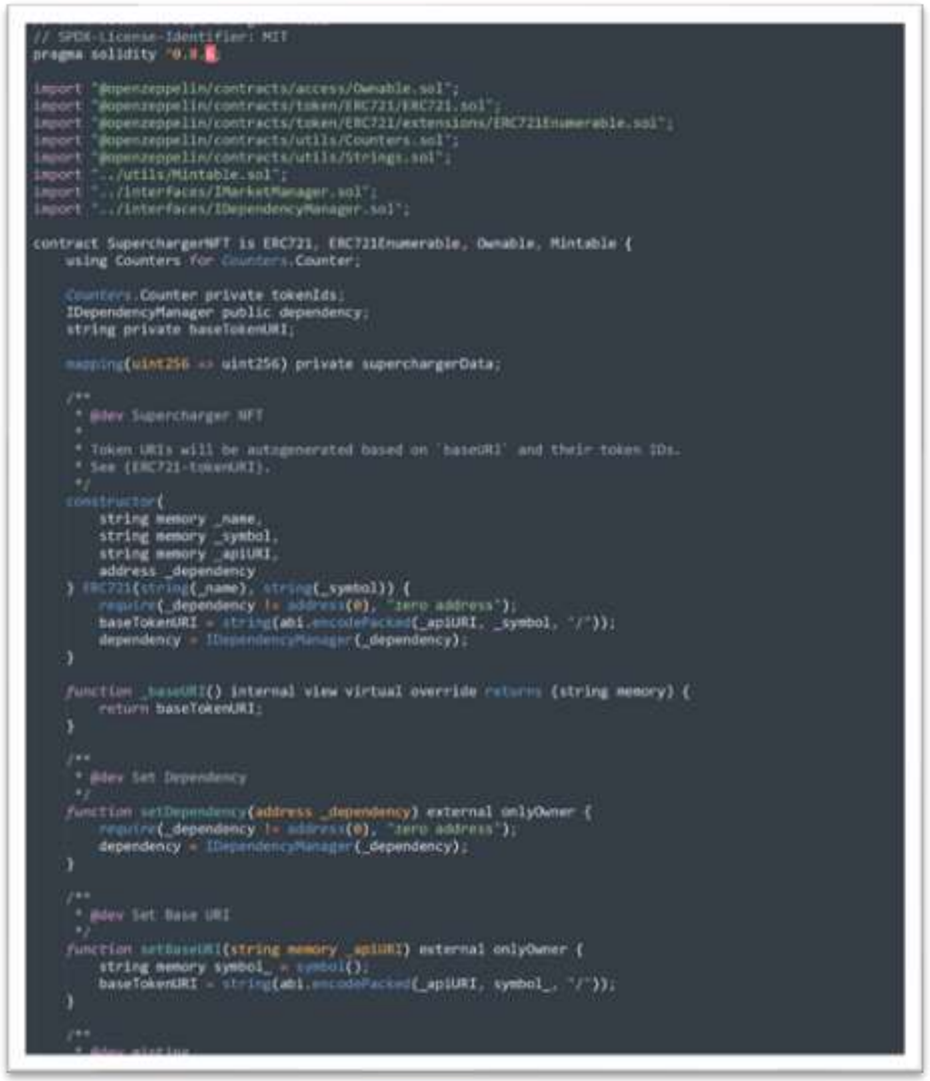
Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

QuantumShards

The vulnerabilities listed above were not found in the token's Smart Contract.

BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on Supercharger NFT (SCHR)



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "@openzeppelin/contracts/utils/Strings.sol";
import "../utils/Mintable.sol";
import "../interfaces/IMarketManager.sol";
import "../interfaces/IDependencyManager.sol";

contract SuperchargerNFT is ERC721, ERC721Enumerable, Ownable, Mintable {
    using Counters for Counters.Counter;

    Counters.Counter private tokenIds;
    IDependencyManager public dependency;
    string private baseTokenURI;

    mapping(uint256 => uint256) private superchargerData;

    /**
     * @dev Supercharger NFT
     *
     * Token URIs will be autogenerated based on `baseURI` and their token IDs.
     * See {ERC721-tokenURI}.
     */
    constructor(
        string memory _name,
        string memory _symbol,
        string memory _apiURI,
        address _dependency
    ) ERC721(string(_name), string(_symbol)) {
        require(_dependency != address(0), "zero address");
        baseTokenURI = string(abi.encodePacked(_apiURI, _symbol, "/"));
        dependency = IDependencyManager(_dependency);
    }

    function _baseURI() internal view virtual override returns (string memory) {
        return baseTokenURI;
    }

    /**
     * @dev Set Dependency
     */
    function setDependency(address _dependency) external onlyOwner {
        require(_dependency != address(0), "zero address");
        dependency = IDependencyManager(_dependency);
    }

    /**
     * @dev Set Base URI
     */
    function setBaseURI(string memory _apiURI) external onlyOwner {
        string memory symbol_ = symbol();
        baseTokenURI = string(abi.encodePacked(_apiURI, symbol_, "/"));
    }

    /**
```

Contract Address
0x1de1a1205e8EAF08fDE233ecA5B5f41cBbc14925

TokenTracker
Supercharger NFT (SCHR)

Contract Creator
0xdc6Ed9A7a2AFa95A2CB920FD50a14A75248E9561

Source Code
Contract Source Code Verified

Contract Name
SuperchargerNFT

Other Settings
default evmVersion

Compiler Version
v0.8.6+commit.11564f7e

Optimization Enabled
Yes with 200 runs

Code is truncated to fit the constraints of this document.
[The code in its entirety can be viewed here.]

The contract code is **verified** on BSCScan.

BEP-20 Contract Code Audit – Vulnerabilities Checked

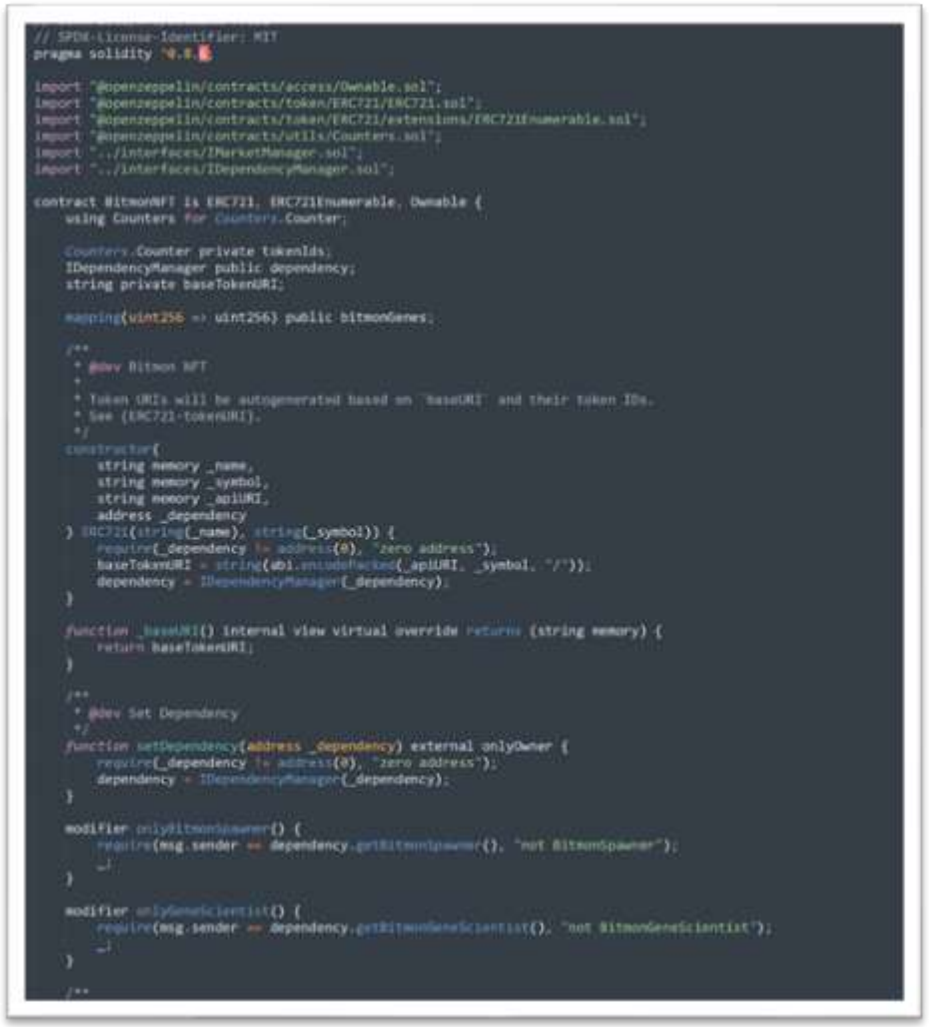
Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

SuperchargerNFT

The vulnerabilities listed above were not found in the token's Smart Contract.

BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on Bitmon NFT (BMON)



```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.6;

import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/token/ERC721/ERC721.sol";
import "@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol";
import "@openzeppelin/contracts/utils/Counters.sol";
import "../interfaces/IMarketManager.sol";
import "../interfaces/IDependencyManager.sol";

contract BitmonNFT is ERC721, ERC721Enumerable, Ownable {
    using Counters for Counters.Counter;

    Counters.Counter private tokenIds;
    IDependencyManager public dependency;
    string private baseTokenURI;

    mapping(uint256 => uint256) public bitmonGenes;

    /**
     * @dev Bitmon NFT
     *
     * Token URIs will be autogenerated based on `baseURI` and their token IDs.
     * See {ERC721-tokenURI}.
     */
    constructor(
        string memory _name,
        string memory _symbol,
        string memory _apiURI,
        address _dependency
    ) ERC721(string(_name), string(_symbol)) {
        require(_dependency != address(0), "zero address");
        baseTokenURI = string(abi.encodePacked(_apiURI, _symbol, "/"));
        dependency = IDependencyManager(_dependency);
    }

    function _baseURI() internal view virtual override returns (string memory) {
        return baseTokenURI;
    }

    /**
     * @dev Set Dependency
     */
    function setDependency(address _dependency) external onlyOwner {
        require(_dependency != address(0), "zero address");
        dependency = IDependencyManager(_dependency);
    }

    modifier onlyBitmonSpawner() {
        require(msg.sender == dependency.getBitmonSpawner(), "not BitmonSpawner");
        _;
    }

    modifier onlyGeneScientist() {
        require(msg.sender == dependency.getBitmonGeneScientist(), "not BitmonGeneScientist");
        _;
    }

    /**
```

Contract Address
0xa9297D069D2d4453491E75E3c765EBF63d116d5C

TokenTracker
Bitmon NFT (BMON)

Contract Creator
0xdc6Ed9A7a2AFa95A2CB920FD50a14A75248E9561

Source Code
Contract Source Code Verified

Contract Name
BitmonNFT

Other Settings
default evmVersion

Compiler Version
v0.8.6+commit.11564f7e

Optimization Enabled
Yes with 200 runs

Code is truncated to fit the constraints of this document.
[The code in its entirety can be viewed here.]

The contract code is **verified** on BSCScan.

BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	AI Scan	Human Review	Result
Compiler Errors	Complete	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	Complete	✓ Low / No Risk
Integer Overflow	Complete	Complete	✓ Low / No Risk
Integer Underflow	Complete	Complete	✓ Low / No Risk
Correct Token Standards Implementation	Complete	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	Complete	✓ Low / No Risk
Deployer Can Access User Funds	Complete	Complete	✓ Low / No Risk

BitmonNFT

The vulnerabilities listed above were not found in the token's Smart Contract.

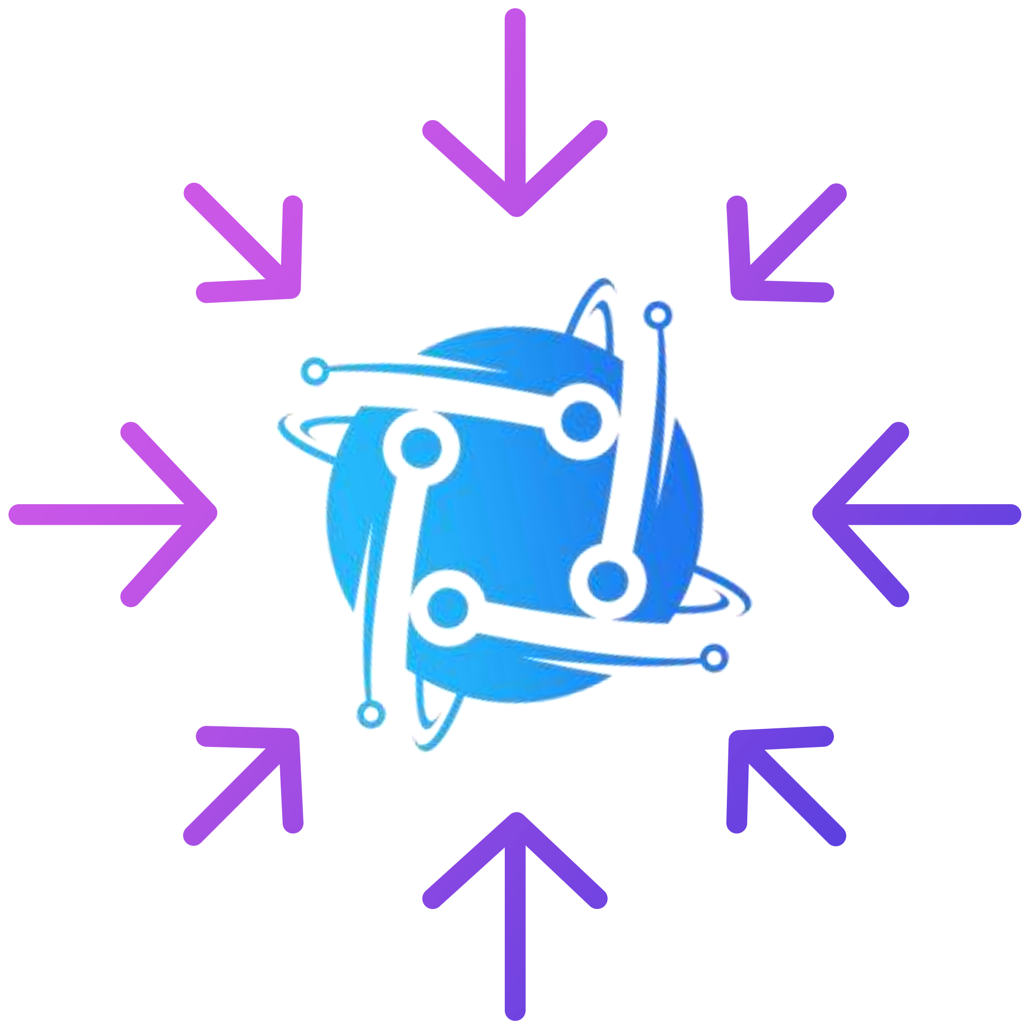
Website Part 1 – Overview
www.bitmonparadise.com

Above images are actual snapshots of the current live website of the project.

Website was registered on 12/04/21, registration expires 12/04/2024.

✓ This meets the 3 year minimum we like to see on new projects.

Website Part 2 – Checklist



- ✓ **Mobile Friendly**
- ✓ **No JavaScript Errors**
- ✓ **Spell Check**
- ✓ **SSL Certificate**

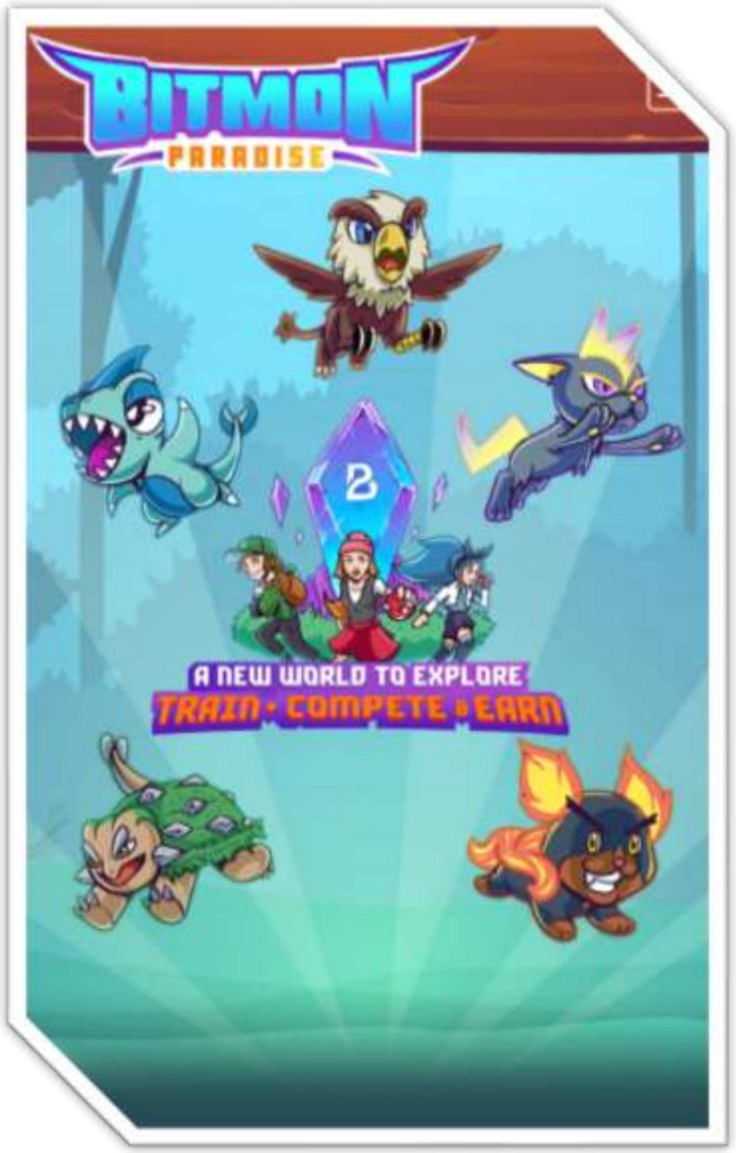
The website contained no JavaScript errors. No typos, or grammatical errors were present, and we found a valid SSL certificate allowing for access via https.

No additional issues were found on the website.

Website Part 3 – Responsive HTML5 & CSS3

No issues were found on the Mobile Friendly check for the website. All elements loaded properly and browser resize was not an issue. The team has put a considerable amount of thought and effort into making sure their website looks great on all screens.

No severe JavaScript errors were found. No issues with loading elements, code, or stylesheets.



Website Part 4 (GWS) – General Web Security

SSL CERTIFICATE

A valid SSL certificate was found. Details are as follows:

Offered to: bitmonparaise.com

Issued by: cPanel, Inc

Valid Until: 06/11/2022

✓

CONTACT EMAIL

A valid contact email was found on the official website. Contact email is listed as shown below:

Contact

support@bitmonparadise.com

✓

SPAM / MALWARE / POPUPS

No malware found

No injected spam found

No internal server errors

No popups found

Domain is marked clean by Google, McAfee, Sucuri Labs, & ESET

✓

Social Media

We were able to locate a variety of Social Media networks for the project.

All links have been conveniently placed below.

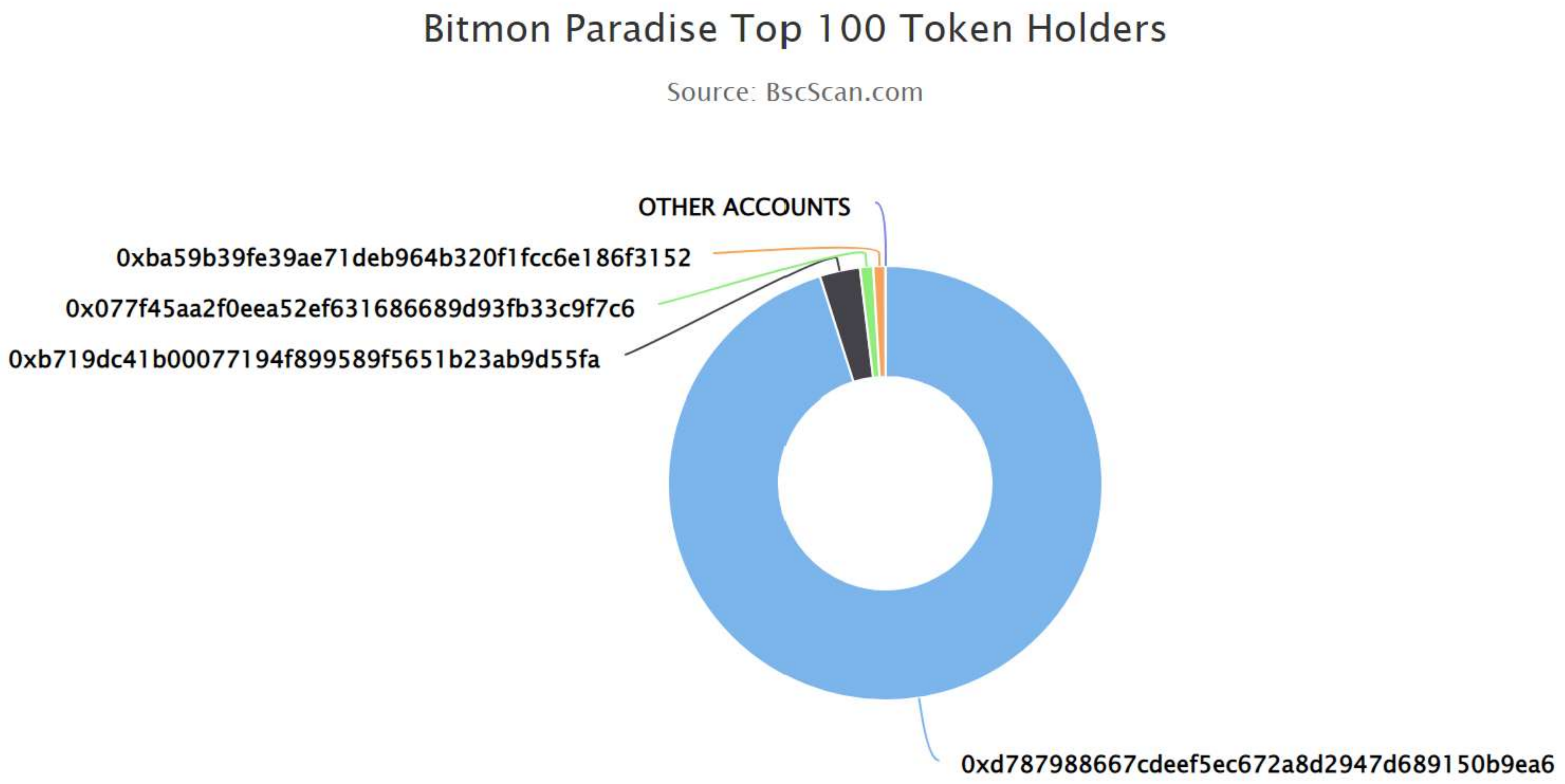
Twitter	Telegram	Reddit	Discord	Facebook	Instagram	Tiktok

✓ **At least 3 social media networks were found.**

Top Token Holders

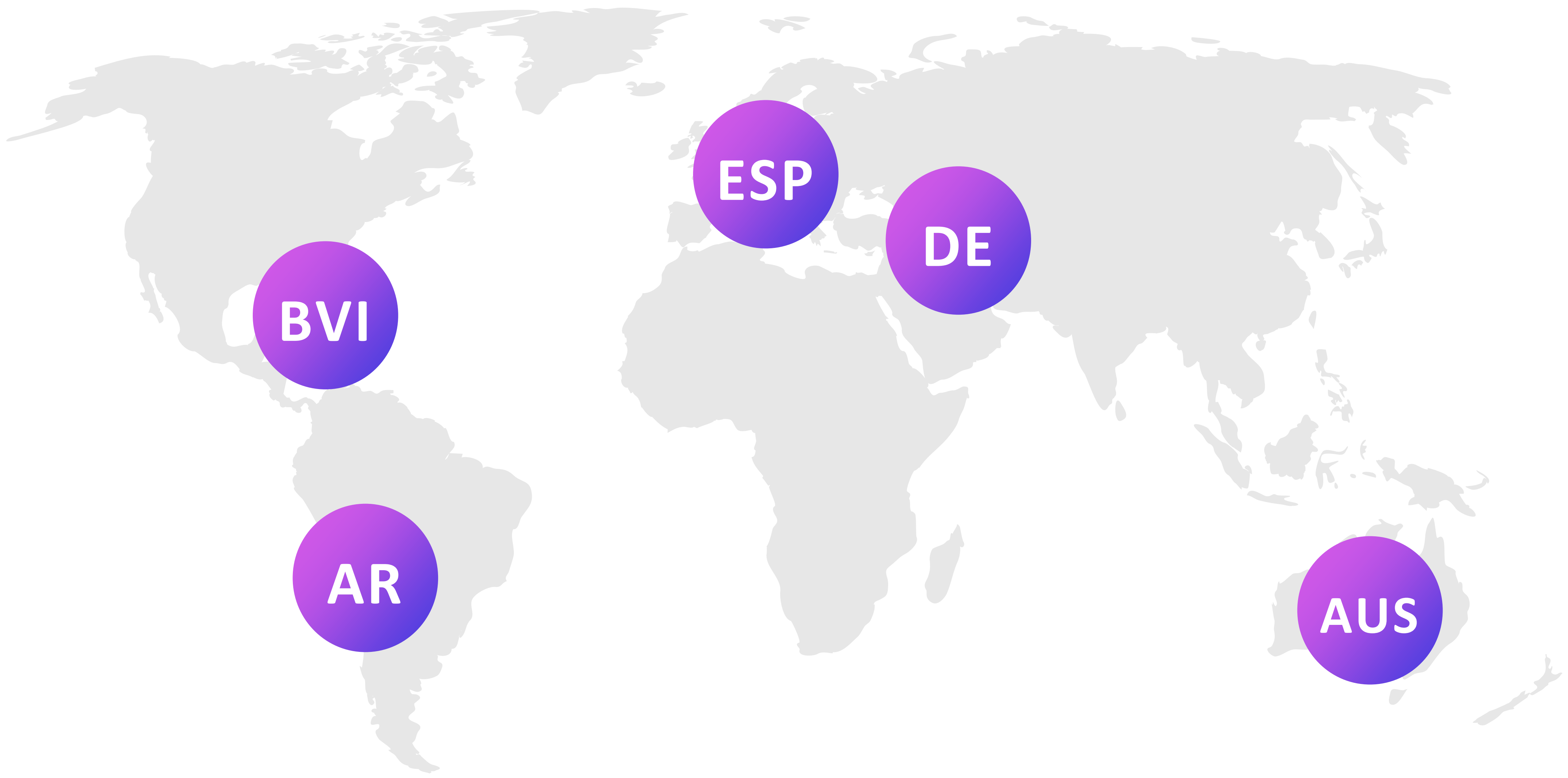
The entire supply was in one wallet at the time of audit. We expect this to change as the project goes through initial distribution phases. Please use the link below to view the most up-to-date holder information.

Click here to view the most up-to-date list of holders

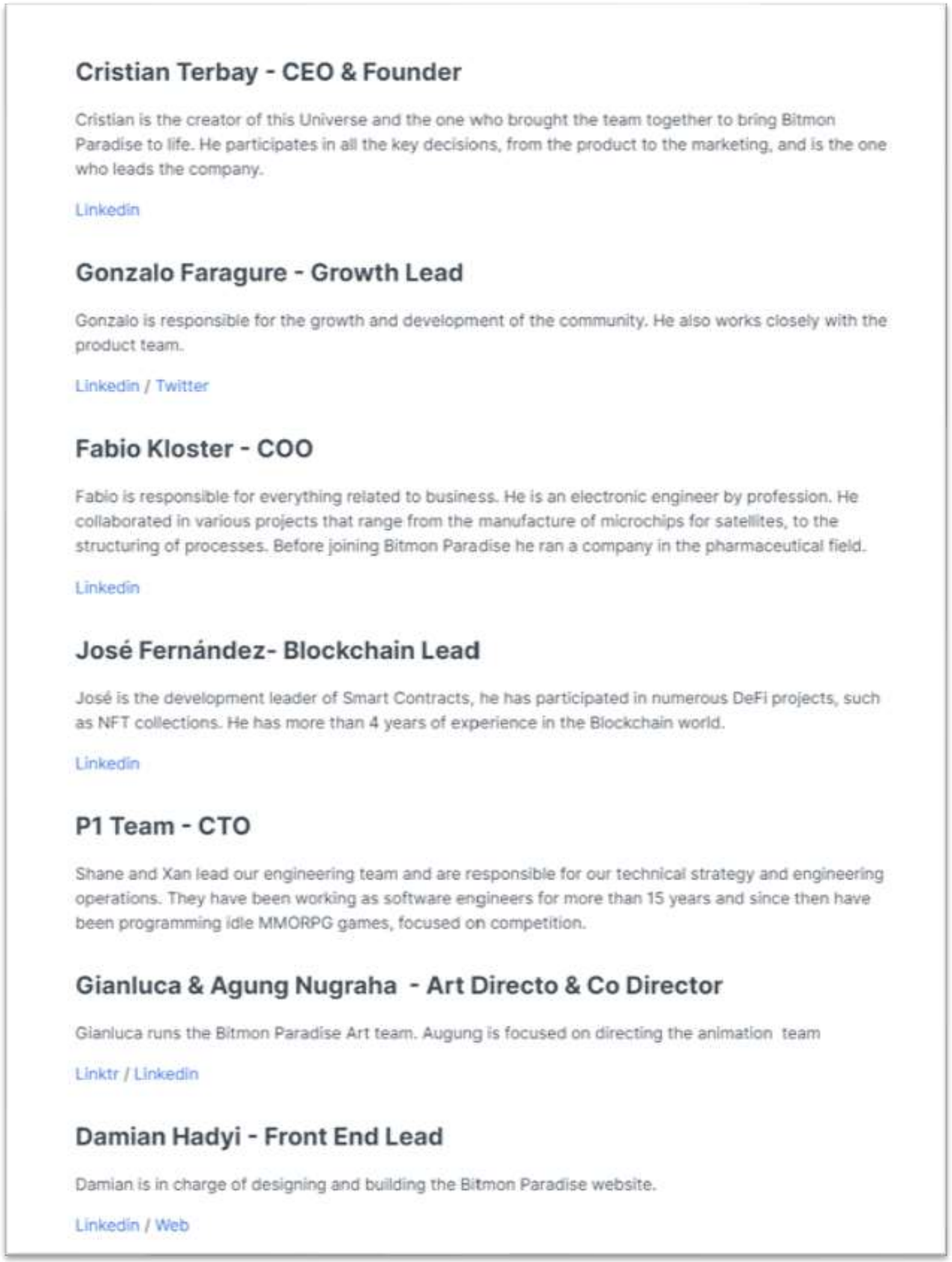


Location Audit

The company was found to be registered in the British Virgin Islands with team members from across the globe shown below.



Team Overview



Cristian Terbay - CEO & Founder

Cristian is the creator of this Universe and the one who brought the team together to bring Bitmon Paradise to life. He participates in all the key decisions, from the product to the marketing, and is the one who leads the company.

Linkedin

Gonzalo Faragure - Growth Lead

Gonzalo is responsible for the growth and development of the community. He also works closely with the product team.

Linkedin / Twitter

Fabio Kloster - COO

Fabio is responsible for everything related to business. He is an electronic engineer by profession. He collaborated in various projects that range from the manufacture of microchips for satellites, to the structuring of processes. Before joining Bitmon Paradise he ran a company in the pharmaceutical field.

Linkedin

José Fernández- Blockchain Lead

José is the development leader of Smart Contracts, he has participated in numerous DeFi projects, such as NFT collections. He has more than 4 years of experience in the Blockchain world.

Linkedin

P1 Team - CTO

Shane and Xan lead our engineering team and are responsible for our technical strategy and engineering operations. They have been working as software engineers for more than 15 years and since then have been programming idle MMORPG games, focused on competition.

Gianluca & Agung Nugraha - Art Directo & Co Director

Gianluca runs the Bitmon Paradise Art team. Augung is focused on directing the animation team

Linktr / Linkedin

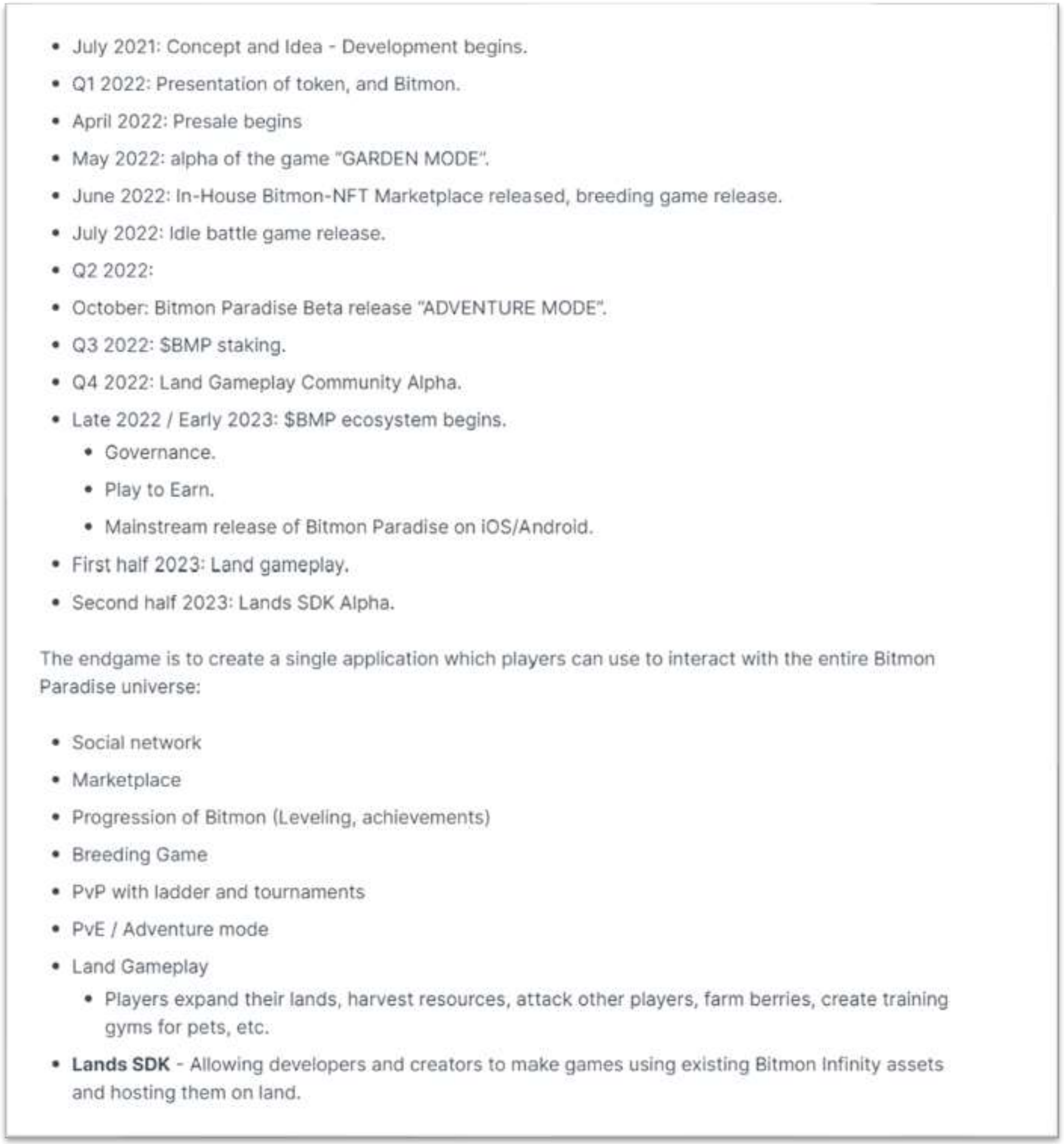
Damian Hadyi - Front End Lead

Damian is in charge of designing and building the Bitmon Paradise website.

Linkedin / Web

Roadmap

A roadmap was found on the official website, we have conveniently placed it on this page for your viewing.



- July 2021: Concept and Idea - Development begins.
- Q1 2022: Presentation of token, and Bitmon.
- April 2022: Presale begins
- May 2022: alpha of the game "GARDEN MODE".
- June 2022: In-House Bitmon-NFT Marketplace released, breeding game release.
- July 2022: Idle battle game release.
- Q2 2022:
- October: Bitmon Paradise Beta release "ADVENTURE MODE".
- Q3 2022: $BMP staking.
- Q4 2022: Land Gameplay Community Alpha.
- Late 2022 / Early 2023: $BMP ecosystem begins.
  - Governance.
  - Play to Earn.
  - Mainstream release of Bitmon Paradise on iOS/Android.
- First half 2023: Land gameplay.
- Second half 2023: Lands SDK Alpha.

The endgame is to create a single application which players can use to interact with the entire Bitmon Paradise universe:

- Social network
- Marketplace
- Progression of Bitmon (Leveling, achievements)
- Breeding Game
- PvP with ladder and tournaments
- PvE / Adventure mode
- Land Gameplay
  - Players expand their lands, harvest resources, attack other players, farm berries, create training gyms for pets, etc.
- **Lands SDK** - Allowing developers and creators to make games using existing Bitmon Infinity assets and hosting them on land.

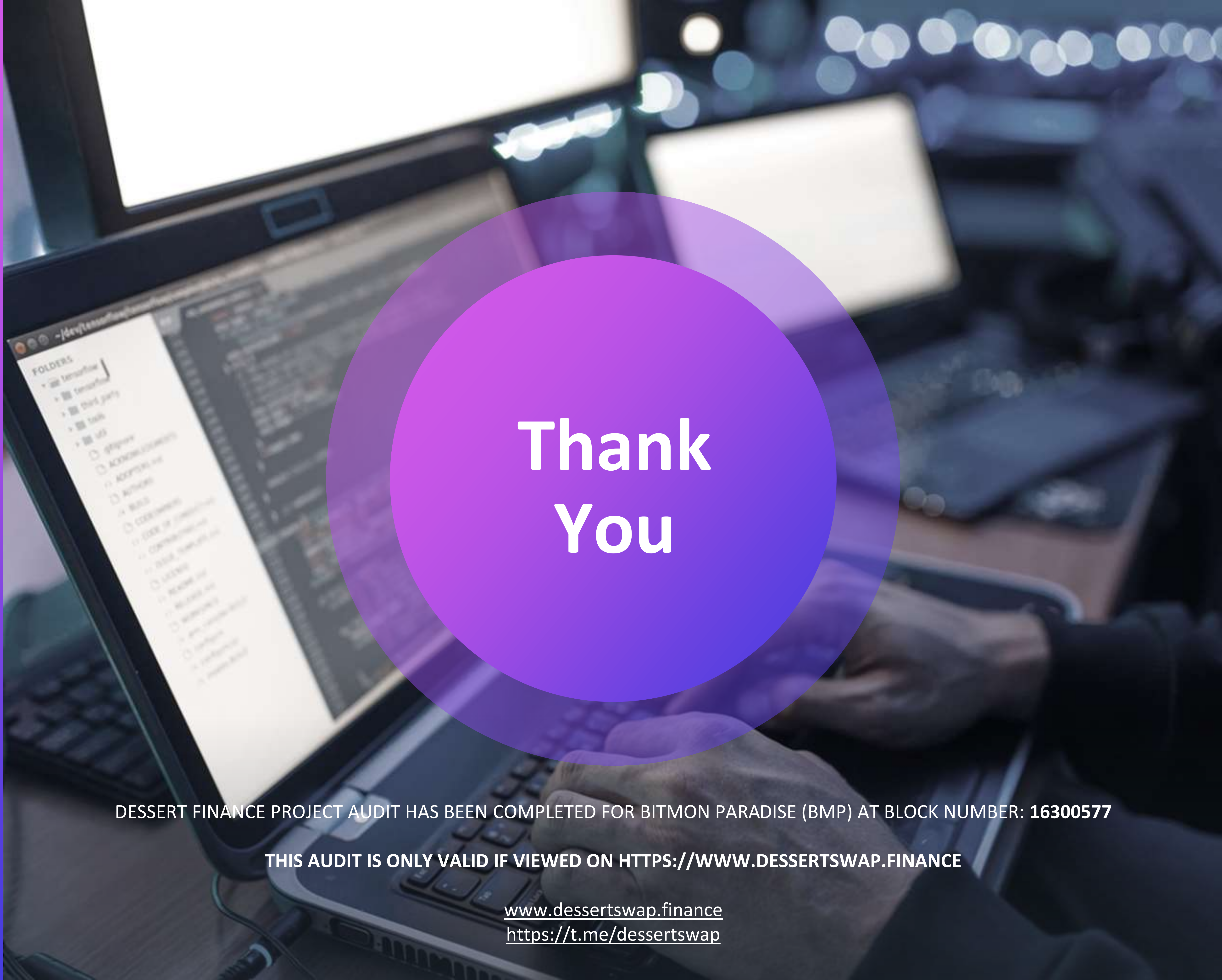
Disclaimer

The opinions expressed in this document are for general informational purposes only and are **not intended to provide specific advice or recommendations for any individual or on any specific investment**. It is only intended to provide education and public knowledge regarding projects. This audit is only applied to the type of auditing specified in this report and the scope of given in the results. Other unknown security vulnerabilities are beyond responsibility. Dessert Finance only issues this report based on the attacks or vulnerabilities that already existed or occurred before the issuance of this report. For the emergence of new attacks or vulnerabilities that exist or occur in the future, Dessert Finance lacks the capability to judge its possible impact on the security status of smart contracts, thus taking no responsibility for them. The smart contract analysis and other contents of this report are based solely on the documents and materials that the contract provider has provided to Dessert Finance or was publicly available before the issuance of this report (issuance of report recorded via block number on cover page), if the documents and materials provided by the contract provider are missing, tampered, deleted, concealed or reflected in a situation that is inconsistent with the actual situation, or if the documents and materials provided are changed after the issuance of this report, Dessert Finance assumes no responsibility for the resulting loss or adverse effects. Due to the technical limitations of any organization, this report conducted by Dessert Finance still has the possibility that the entire risk cannot be completely detected. Dessert Finance disclaims any liability for the resulting losses.

Dessert Finance provides no guarantees against the sale of team tokens or the removal of liquidity by the project audited in this document. Even projects with a low risk score have been known to pull liquidity, sell all team tokens, or exit-scam. Please exercise caution when dealing with any cryptocurrency related platforms.

The final interpretation of this statement belongs to Dessert Finance.

Dessert Finance highly advises against using cryptocurrencies as speculative investments and they should be used solely for the utility they aim to provide.



Thank You

DESSERT FINANCE PROJECT AUDIT HAS BEEN COMPLETED FOR BITMON PARADISE (BMP) AT BLOCK NUMBER: **16300577**

THIS AUDIT IS ONLY VALID IF VIEWED ON HTTPS://WWW.DESSERTSWAP.FINANCE

www.dessertswap.finance
https://t.me/dessertswap