

**DESSERT
FINANCE**



**OGPEPE (OGPEPE)
LIGHT AUDIT**

Performed by Dessert Finance

PERFORMED BY DESSERT FINANCE
FOR CONTRACT ADDRESS: 0x01a7211a611A4B501EBa570f3d71405B42300279

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.

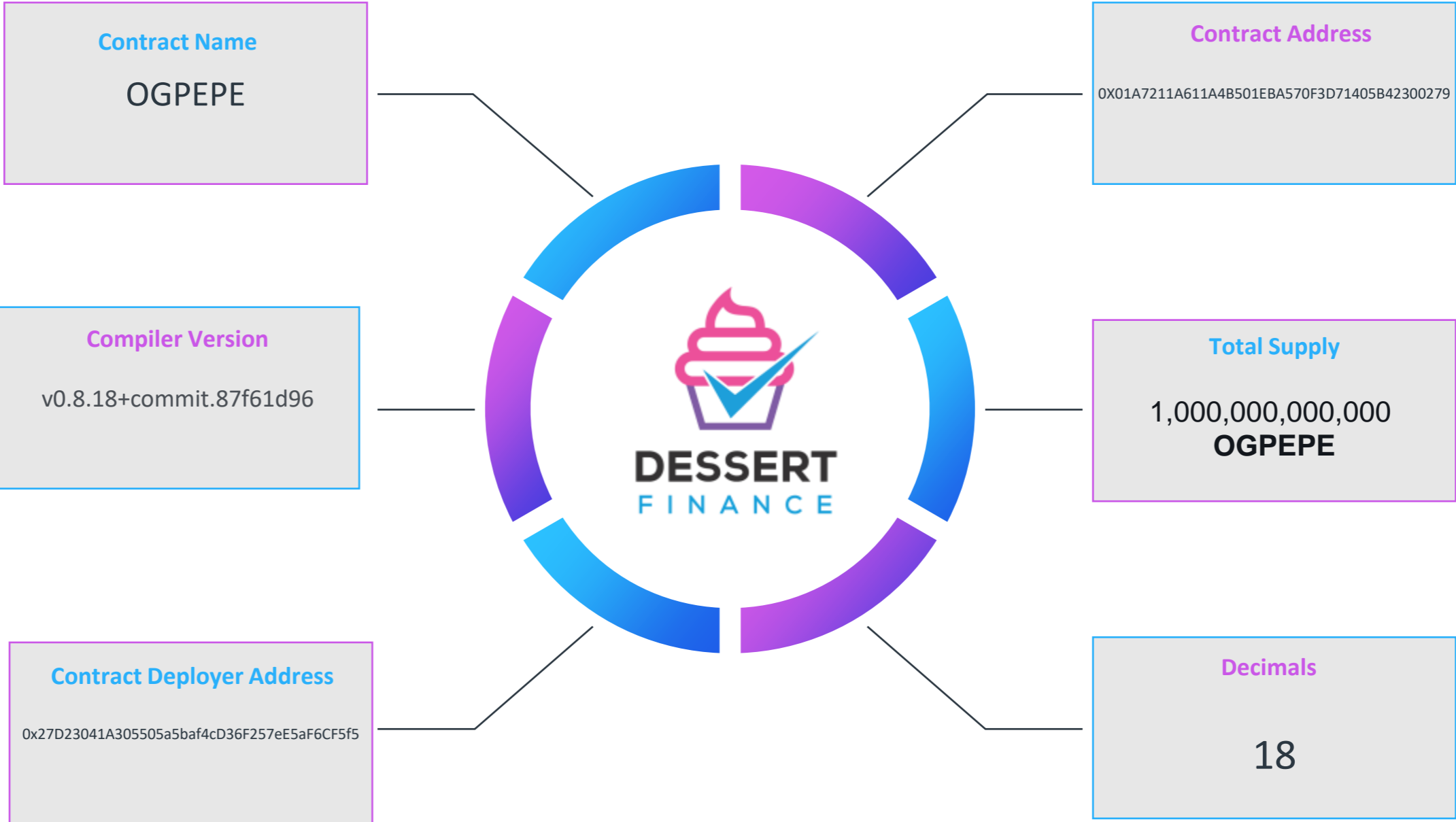
Dessert Finance in no way takes responsibility for any losses, nor does Dessert Finance encourage any speculative investments. The information provided in this audit is for information purposes only and should not be considered investment advice. Dessert Finance does not endorse, recommend, support, 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



1. Contract Code Audit – Token Overview
2. BEP-20 Contract Code Audit – Overview
3. BEP-20 Contract Code Audit – Vulnerabilities Checked
4. Mint Function
5. Ownership
6. Owner Accessible Functions
7. Disclaimers

Contract Code Audit – Token Overview



BEP-20 Contract Code Audit – Overview

Dessert Finance was commissioned to perform an audit on OGPEPE (OGPEPE)

```
// Telegram: https://t.me/OGPepeAI
pragma solidity ^0.8.18;

abstract contract Context {
    function msgSender() internal view virtual returns (address) {
        return msg.sender;
    }

    function msgData() internal view virtual returns (bytes calldata) {
        this; // silence state mutability warning without generating bytecode - see https://github.com/ethereum/solidity/issues/2691
        return msg.data;
    }
}

interface IUniswapV2Pair {
    event Approval(address indexed owner, address indexed spender, uint value);
    event Transfer(address indexed from, address indexed to, uint value);

    function name() external pure returns (string memory);
    function symbol() external pure returns (string memory);
    function decimals() external pure returns (uint8);
    function totalSupply() external view returns (uint);
    function balanceOf(address owner) external view returns (uint);
    function allowance(address owner, address spender) external view returns (uint);

    function approve(address spender, uint value) external returns (bool);
    function transfer(address to, uint value) external returns (bool);
    function transferFrom(address from, address to, uint value) external returns (bool);

    function DOMAIN_SEPARATOR() external view returns (bytes32);
    function PERMIT_TYPEHASH() external pure returns (bytes32);
    function nonces(address owner) external view returns (uint);

    function permit(address owner, address spender, uint value, uint deadline, uint8 v, bytes32 r, bytes32 s) external;

    event Mint(address indexed sender, uint amount0, uint amount1);
    event Burn(address indexed sender, uint amount0, uint amount1, address indexed to);
    event Swap(
        address indexed sender,
        uint amount0In,
        uint amount1In,
        uint amount0Out,
        uint amount1Out,
        address indexed to
    );
    event Sync(uint112 reserve0, uint112 reserve1);

    function MINIMUM_LIQUIDITY() external pure returns (uint);
    function factory() external view returns (address);
    function token0() external view returns (address);
    function token1() external view returns (address);
    function getReserve0() external view returns (uint112 reserve0, uint112 reserve1, uint32 blockTimestampLast);
    function price0CumulativeLast() external view returns (uint);
    function price1CumulativeLast() external view returns (uint);
    function kLast() external view returns (uint);

    function wot(address to) external returns (uint liquidity);
    function burn(address to) external returns (uint amount0, uint amount1);
}
```

Contract Address

0x01a7211a611A4B501EBa570f3d71405B42300279

TokenTracker

OGPEPE (OGPEPE)

Contract Creator

0x27d23041a305505a5baf4cd36f257ee5af6cf5f5

Source Code

Contract Source Code Verified

Contract Name

OGPEPE

Other Settings

default evmVersion, MIT

Compiler Version

v0.8.18+commit.87f61d96

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.](#)

BEP-20 Contract Code Audit – Vulnerabilities Checked

Vulnerability Tested	Scan	Result
Compiler Errors	Complete	✓ Low / No Risk
Outdated Compiler Version	Complete	✓ Low / No Risk
Integer Overflow	Complete	✓ Low / No Risk
Integer Underflow	Complete	✓ Low / No Risk
Floating Pragma	Complete	✓ Low / No Risk
Timestamp Dependency for Crucial Functions	Complete	✓ Low / No Risk
Exposed _Transfer Function	Complete	✓ Low / No Risk
Transaction-Ordering Dependency	Complete	✓ Low / No Risk
Unchecked Call Return Variable	Complete	✓ Low / No Risk
Use of Deprecated Functions	Complete	✓ Low / No Risk
Unprotected SELFDESTRUCT Instruction	Complete	✓ Low / No Risk
State Variable Default Visibility	Complete	✓ Low / No Risk

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

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

Contract Code Audit – Mint Functions

This Contract Cannot Mint New OGPEPE 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.

Contract Code Audit – Contract Ownership

Contract Ownership has not been renounced at the time of Audit



The contract ownership is not currently renounced.

We have placed the contract owner address below for your viewing:

[0x27d23041a305505a5baf4cd36f257ee5af6cf55](https://www.etherbase.io/etherscan/address/0x27d23041a305505a5baf4cd36f257ee5af6cf55)

The address above has authority over the ownable functions within the contract.

This allows the owner to call certain functions within the contract. Any compromise to the owner wallet may allow these privileges to be exploited.

We recommend:

- Establishing a Time-Lock with reasonable latency
- Assignment of privileged roles to multi-signature wallets

Contract Code Audit – Owner Accessible Functions

Function Name	Parameters	Visibility	Audit Notes
renounceOwnership		public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
transferOwnership	address newOwner	public virtual	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
enableTrading		external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
removeLimits		external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
disableTransferDelay		external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateSwapTokensAtAmount	uint256 newAmount	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateMaxTxnAmount	uint256 newNum	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateMaxWalletAmount	uint256 newNum	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
excludeFromMaxTransaction	address updAds, bool isEx	public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateSwapEnabled	bool enabled	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateBuyFees	uint256 _marketingFee, uint256 _liquidityFee, uint256 _devFee	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateSellFees	uint256 _marketingFee, uint256 _liquidityFee, uint256 _devFee	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
excludeFromFees	address account, bool excluded	public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
setAutomatedMarketMakerPair	address pair, bool value	public	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateMarketingWallet	address newMarketingWallet	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
updateDevWallet	address newWallet	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.
setAutoLPBurnFrequency	uint256 _frequencyInSeconds, uint256 _percent, bool _Enabled	external	onlyOwner modifier is detected. Owner can call this function if the contract is not renounced.

The functions listed above can be called by the contract owner.

If contract ownership has been renounced there is no way for the above listed functions to be called.

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 LIGHT AUDIT HAS BEEN COMPLETED FOR OGPEPE (OGPEPE)
THIS AUDIT IS ONLY VALID IF VIEWED ON [HTTPS://WWW.DSSERTSWAP.FINANCE](https://www.dessertswap.finance)

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