**# Whitepaper: gammaStaking Contract**

# **## Abstract**

**The gammaStaking contract is a decentralized staking and rewards distribution platform built on the Ethereum blockchain. It enables users to stake a specific ERC20 token and receive rewards in return. The contract also integrates with an ERC721 token, allowing users to stake specific NFTs for additional benefits.**

**## Table of Contents**

**1. \*\*Introduction\*\***

**- 1.1 Overview**

**- 1.2 Purpose**

**2. \*\*Contract Features\*\***

**- 2.1 Staking Mechanism**

**- 2.2 Reward Distribution**

**- 2.3 ERC20 Integration**

**- 2.4 ERC721 Integration**

**- 2.5 Access Control**

**- 2.6 ReentrancyGuard and Pausable**

**- 2.7 Minting Rewards**

**3. \*\*Contract Parameters\*\***

**- 3.1 Token Details**

**- 3.2 Staking Requirements**

**- 3.3 Rewards Configuration**

**- 3.4 Compounding Frequency**

**4. \*\*Contract Functions\*\***

**- 4.1 buyTokens**

**- 4.2 sellTokens**

**- 4.3 DAOstake**

**- 4.4 DAOunstake**

**- 4.5 setMinersReward**

**- 4.6 setUsersReward**

**- 4.7 setRecip2Reward**

**- 4.8 \_mintMinerReward**

**- 4.9 setRewardsPerHour**

**- 4.10 setMinStake**

**- 4.11 deposit**

**- 4.12 stakeRewards**

**- 4.13 claimRewards**

**- 4.14 withdraw**

**- 4.15 withdrawAll**

**- 4.16 getDepositInfo**

**- 4.17 compoundRewardsTimer**

**- 4.18 calculateRewards**

**- 4.19 pause**

**- 4.20 unpause**

**- 4.21 \_beforeTokenTransfer**

**- 4.22 \_afterTokenTransfer**

**- 4.23 \_mint**

**- 4.24 \_burn**

**- 4.25 mint**

**- 4.26 reward\_mint**

**5. \*\*Conclusion\*\***

**- 5.1 Future Developments**

**- 5.2 Contact Information**

**## 1. Introduction**

**### 1.1 Overview**

**The gammaStaking contract is a versatile staking and rewards distribution platform designed to incentivize users for holding and staking ERC20 tokens. Additionally, it integrates with an ERC721 token (DAO token) to provide enhanced benefits for stakers.**

**### 1.2 Purpose**

**The primary purpose of the gammaStaking contract is to create an ecosystem where users can stake tokens and receive rewards based on their staked amount and duration. The contract also introduces features to stake specific NFTs (DAO tokens), unlocking additional rewards.**

**## 2. Contract Features**

**### 2.1 Staking Mechanism**

**Users can stake ERC20 tokens, and the contract calculates rewards based on the staked amount and time duration. Staked tokens are burned to represent the locked value.**

**### 2.2 Reward Distribution**

**The contract distributes rewards to stakers based on a configurable rewards-per-hour mechanism. Rewards are calculated continuously and can be claimed or compounded.**

**### 2.3 ERC20 Integration**

**The contract extends the ERC20 standard, allowing seamless interaction with the Ethereum ecosystem and enabling users to buy, sell, and transfer staking tokens.**

**### 2.4 ERC721 Integration**

**The contract integrates with an ERC721 token (DAO token). Users can stake these NFTs for additional rewards and benefits.**

**### 2.5 Access Control**

**Access control is implemented using OpenZeppelin's AccessControl module, providing role-based permissions for contract administration.**

# **### 2.6 ReentrancyGuard and Pausable**

**To enhance security, the contract includes ReentrancyGuard and Pausable features, preventing reentrancy attacks and allowing the contract owner to pause and unpause certain functionalities.**

**### 2.7 Minting Rewards**

**The contract mints new tokens as rewards for stakers, including rewards for miners, users, and a specific recipient.**

**## 3. Contract Parameters**

**### 3.1 Token Details**

**The contract operates with a specific ERC20 token, with a maximum supply of 55,555,555 tokens.**

**### 3.2 Staking Requirements**

**Users are required to stake a minimum amount (minStake) to participate in the staking program.**

**### 3.3 Rewards Configuration**

**The contract allows the owner to configure rewards distribution, including rewards per hour and various reward allocations.**

**### 3.4 Compounding Frequency**

**Compounding of rewards is limited to a specific frequency (compoundFreq), ensuring fair distribution and preventing abuse.**

**## 4. Contract Functions**

**### 4.1 buyTokens**

**Allows users to purchase staking tokens by sending ETH to the contract.**

**### 4.2 sellTokens**

**Enables users to sell staking tokens and receive ETH in return.**

**### 4.3 DAOstake**

**Allows users to stake DAO tokens, receiving staking tokens and rewards.**

**### 4.4 DAOunstake**

**Allows users to unstake DAO tokens after a specified minimum time.**

**### 4.5 setMinersReward**

**Allows the contract owner to set the reward for miners.**

**### 4.6 setUsersReward**

**Allows the contract owner to set the reward for users.**

**### 4.7 setRecip2Reward**

**Allows the contract owner to set the reward for a specific recipient.**

**### 4.8 \_mintMinerReward**

**Mints rewards for miners, users, and a specific recipient.**

**### 4.9 setRewardsPerHour**

**Allows the contract owner to set the rewards-per-hour parameter.**

**### 4.10 setMinStake**

**Allows the contract owner to set the minimum stake required.**

**### 4.11 deposit**

**Allows users to deposit staking tokens, initiating or updating their staking position.**

**### 4.12 stakeRewards**

**Allows users to compound their rewards by staking them again.**

**### 4.13 claimRewards**

**Allows users to claim their accumulated rewards.**

**### 4.14 withdraw**

**Allows users to withdraw a specified amount of staked tokens.**

**### 4.15 withdrawAll**

**Allows users to withdraw all staked tokens and rewards.**

**### 4.16 getDepositInfo**

**Returns information about a user's current staking position and rewards.**

**### 4.17 compoundRewardsTimer**

**Returns the remaining time until a user can compound their rewards.**

**### 4.18 calculateRewards**

**Calculates the rewards accrued by a user since their last update.**

**### 4.19 pause**

**Pauses certain contract functionalities, preventing further actions.**

**### 4.20 unpause**

**Resumes contract functionalities after being paused.**

**### 4.21 \_beforeTokenTransfer**

**Overrides the ERC20 function to include additional checks before token transfers.**

**### 4.22 \_afterTokenTransfer**

**Overrides the ERC20 function to include additional actions after token transfers.**

**### 4.23 \_mint**

**Overrides the ERC20 function to include additional actions when minting tokens.**

**### 4.24 \_burn**

**Overrides the ERC20 function to include additional actions when burning tokens**

**.**

**### 4.25 mint**

**Allows the contract owner to mint additional tokens.**

**### 4.26 reward\_mint**

**Allows a specific role (MINTER\_ROLE) to mint additional reward tokens.**

# **## 5. Conclusion**

**The gammaStaking contract provides a robust and flexible solution for staking ERC20 tokens, integrating with ERC721 tokens for enhanced benefits. Users can participate in the staking program, receive rewards, and compound their earnings. The contract's modular design and configurable parameters make it adaptable to various use cases within the decentralized finance (DeFi) space.**

**### 5.1 Future Developments**

**Future developments may include additional features, optimizations, and integrations based on community feedback and evolving industry standards.**

**### 5.2 Contact Information**

**For inquiries and support, please contact the contract owner at [owner@example.com].**

**---**

**\*\*Note:\*\* This whitepaper serves as documentation for the gammaStaking contract and provides an overview of its features and functionalities. It is not financial advice, and users are encouraged to review the contract code and consult with experts before participating.**

**### 1. \*\*Reward Calculation\*\***

**The rewards are calculated based on the staked amount and the time the tokens have been staked. The formula for calculating rewards is as follows:**

**```solidity**

**function calculateRewards(address \_staker) internal view returns (uint256 rewards) {**

**return (((((block.timestamp - stakers[\_staker].timeOfLastUpdate) \* stakers[\_staker].deposited) \* rewardsPerHour) / 3600) / 10000000);**

**}**

**```**

**Here:**

**- `block.timestamp - stakers[\_staker].timeOfLastUpdate` represents the time elapsed since the last update.**

**- `stakers[\_staker].deposited` represents the amount of tokens staked.**

**- `rewardsPerHour` is the configured reward rate per hour.**

**- The division by `3600` converts the time from seconds to hours.**

**- The final division by `10000000` is used to adjust the precision and obtain the reward in the token's smallest unit.**

**### 2. \*\*Compound Rewards Timer\*\***

**The function `compoundRewardsTimer` calculates the time remaining until a user can compound their rewards. The formula is:**

**```solidity**

**function compoundRewardsTimer(address \_user) public view returns (uint256 \_timer) {**

**if (stakers[\_user].timeOfLastUpdate + compoundFreq <= block.timestamp) {**

**return 0;**

**} else {**

**return (stakers[\_user].timeOfLastUpdate + compoundFreq) - block.timestamp;**

**}**

**}**

**```**

**Here:**

**- `stakers[\_user].timeOfLastUpdate + compoundFreq` represents the next allowed compound time.**

**- `block.timestamp` is the current time.**

**- The result is the remaining time until compounding is allowed.**

**### 3. \*\*Buy Tokens\*\***

**The `buyTokens` function calculates the amount of tokens a user receives for a given amount of ETH:**

**```solidity**

**function buyTokens() public payable returns (uint256 tokenAmount) {**

**// ...**

**uint256 amountToBuy = msg.value \* tokensPerETH;**

**// ...}**

**```**

**Here:**

**- `msg.value` is the amount of ETH sent by the user.**

**- `tokensPerETH` is the rate at which tokens are exchanged for ETH.**

**### 4. \*\*Staking Rewards Minting\*\***

**The `\_mintMinerReward` function mints rewards for miners, users, and a specific recipient:**

**```solidity**

**function \_mintMinerReward() internal {**

**\_mint(block.coinbase, minersReward);**

**\_mint(0x26451fb8544613382934F5E761f94ac162bcD9c7, recip2Reward);**

**\_mint(msg.sender, usersReward);**

**}**

**```**

**Here:**

**- `\_mint` is the internal function to mint new tokens.**

**- `block.coinbase` is the miner's address, receiving the miner's reward.**

**- `0x26451fb8544613382934F5E761f94ac162bcD9c7` is a specific recipient's address.**

**- `msg.sender` is the user's address, receiving the user's reward.**

**These mathematical operations are crucial for the proper functioning of the `gammaStaking` contract, ensuring fair and accurate rewards distribution based on staking activities and configured parameters.**