PRESALE

Problem:

- **Funding**: Most presale platforms require funding in either crypto assets or the unlaunched token. If tokens are required, the investor base will experience extra sell pressure since giving tokens to the presale platform is essentially inflation. If crypto assets are required, it raises a barrier of entry to either suppress the existing marketing fund or cancel an unfunded launch.

- **Hidden code**: Hidden code prevents forking and bot manipulation so most crypto projects hide their code. However, there can be exploitable functions within the smart contract either purposefully to steal funds or accidently subject to hacks which will destroy the foundation of the token.

Solution:

- **Free to launch**: To fix the barrier to entry that funding causes, starting a presale will be free. Profits will be acquired if liquidity is unlocked prior to a year.

- **Open-source code**: To fix possible exploits, the presale code will be viewable to anyone and with the use of comments, should be legible to the average crypto investor.

Pricing:

- **Chart**: Crypto investors often look at charts to determine if a project is undervalued or overvalued. The launch price is important to the shape of the chart. By strengthening the liquidity instead of inflating the starting price, the chart shape will tend to point upwards.

- **Liquidity strength**: The price number will be automatically generated based on number of tokens given to the presale, number of native coins raised by the presale, and liquidity ratio. Without the ability to set the price, the strength of the liquidity pool is ensured.

    - **High price**: For example, setting the price too high will reduce the size of the liquidity pool which increases the volatility of price. This will provide a highly profitable exit for the fastest sellers (bots) after the presale is complete, reducing the price and harming the token long term.

    - **Low price**: Setting the price too low will increase the size of liquidity and reduce the ability to move the price. In the altcoin space, price movement is important to maintain investor interest as people want to make money.

- **Liquidity ratio**: The liquidity to funding ratio is set at the creation of presale (50% - 100% into liquidity). Toolbelt will set the ratio to 80% meaning 20% will be sent to the marketing fund and the rest will be sent to liquidity.
    - **Funding**: Funding can be used for marketing, team payment, or anything that the creators of the presale deem necessary. This provides an opportunity for unfunded projects to have a competitive chance.
    - **Price**: Since purchasing power cannot be created for free, the presale users will provide the funding. This means the presale users tokens will be worth the liquidity ratio times the number of coins they invested. By keeping the launch price under the profit mark, exiting the presale at the moment of launch (bot attacks) will be unprofitable.

Functions and Events:

- **newSale**: Creates new presale.
    - **Requirements**: 1 presale per token, swap's core token must be wrapped native coin (WBNB), presale duration less than 1 month, percentIntoLiquidity between 50% and 100%.
    - **Event**: NewSale(address indexed tokenSale, uint indexed percentIntoLiquidity, uint duration);

- **joinSale**: Invest in a presale.
    - **Requirements**: Presale is ongoing, user has funds.
    - **Event**: JoinSale(address indexed tokenSale, address indexed wallet, uint buyPower);

- **endSale**: Complete presale and launch token.
    - **Requirements**: Presale is ongoing, duration is complete.
    - **Event**: EndSale(address indexed tokenSale, uint weiToLiquidity, uint weiToWallet);

- **claimSale**: Collect tokens after investing in a presale.
    - **Requirements**: Presale is complete, user joined presale and didn't claim yet.
    - **Event**: ClaimSale(address indexed tokenSale, address indexed wallet, uint tokensClaimed);

- **unlockLiquidity**: Sends liquidity to creator of presale. Unlocking liquidity prior to a year will result in a penalty. Toolbelt is incapable of unlocking prior to a year.
    - **Penalty**: Thieves are disincentivized from using this presale platform. Under certain circumstances like hacks or migration, liquidity might need to be unlocked prior to a year, so the number of coins penalized will be dynamic. The tokens in liquidity will be burned and untouchable inside the presale smart contract. The coin penalty will be automatically swapped to Toolbelt tokes and sent to stakers.
    - **Fee**: If lock time is complete, unlocking liquidity will result in a 1% fee given directly to stakers. If lock time is incomplete, fees will be calculated with the equation below.
    - **Calculation**: retrieveableCoins = liquidityCoins * (timeLocked / 1year) **2
    - **Requirements**: Liquidity exists, user created presale.
    - **Event**: UnlockLiquidity(address indexed tokenSale, uint weiToWallet, uint weiFee);

Token incentives:

- **From**: Unlocking liquidity will result in a 1% fee. Unlocking prior to a year depends on a curve up to 100% fee.
- **To**: 100% to staking rewards.

STAKING

Problem:

- **Zero-sum**: A zero-sum game is where nothing is added or taken away from the system. Most tokens play this game where investors are buying and selling money amongst each other without generating any new money. This leaves the investor base in a cycle of fighting over the same money.

- **Revenue absence**: Some tokens have staking protocol that reappropriate tokens from the traders to the stakers. This incentivizes long-term holding, but without revenue, funds used by marketing and team salaries will suck the value out of the token.

- **Impermanence**: Many meme tokens follow the same trend. Hype leads to a pump and the absence of utility leads to a dump. The only thing that keeps meme tokens alive is the possibility of a higher price from hype alone.

Solution:

- **Revenue**: With the addition of revenue streams, the investor base will not be playing a zero-sum game; rather profiting from the success of the company. This value addition allows the opportunity of profit even if the investor bought the all-time high. Dapp profit will be sent directly to the staking protocol where the long-term investors reap the benefits.

- **Longevity**: With utilities that solve problems, positive buy pressure from revenue streams, and incentives to buy and stake the token, Toolbelt is built to last.

Functions and Events:

- **stake**: Adds to stake position and claims rewards.
    - **Requirements**: User has enough Toolbelt tokens.
    - **Event**: Stake(address indexed wallet, uint weiTotal);
- **unstake**: Removes entire stake position and claims rewards. Unstaking prior to a year will result in a token penalty.
    - **Penalty**: Staking is built to reward long term investors with revenue. Locking tokens behind a penalty ensures that stakers are long term, but also gives opportunity to exit in case of emergency.
    - **Calculation**: tokenPayout = tokensStaked * stakeTime / 1year;

    - **Requirements**: User has enough Toolbelt tokens.
    - **Event**: Stake(address indexed wallet, uint weiTotal);
- **claim**: Tokens are removed from the rewards pool and added to the user's wallet.
    - **Rewards**: Rewards will fluctuate based on others' claims. Since every taxed transfer will create rewards, the reward calculation is built to be gas efficient rather than conserving individual rewards. There will be claiming strategies invented by stakers to maximize rewards.
    - **Requirements**: User has rewards.
    - **Event**: Stake(address indexed wallet, uint weiTotal);
- **distributeRevenue**: Adds rewards to the total staking pool. Automatically called on taxed transfers and other dapp functions. Can be used to donate tokens to stakers.
    - **Requirements**: Sender has enough funds.

EVENTS

Problem:

- **Hidden**: Events can be lost within the deep array of blockchain interactions. Missing out on valuable information in the worst case can result in catastrophic changes. For example, the Toolbelt locker requires a vote to make a change to the smart contract. This vote could alter the cost of creating a new vote so high, that only the top holders will be able to initiate changes to the smart contract. By disrupting the balance of power, people may become upset and sell their shares.

Solution:

- **Tracking**: By enhancing the ability to track events, users can strengthen their information about smart contract changes. For instance, using the Toolbelt locker system, users can see exactly when a vote has been initiated. This ensures the opportunity to vote and alter the smart contract in the user's favor.

Example:

- Find the contract address that the event exists on.
    - 0x97AF9b7d2bff52b9e0C9C4B437A148A56658e763

- Find the name of the event to track.
    - CreateDelta

- List the parameter types of the event with separated by commas, no spaces, and an underscore if the parameter is indexed. Must be exact to work.
    - address_indexed,address_indexed,uint_indexed

- List the parameter names. This is user preference as the names will be displayed for tracking. No spaces in between commas.
    - Voting token,Contract address,ID

- List filter values to track specific types of events. Since there are 3 parameters in the example function, there's up to 3 inputs required. Leave it blank for tracking every event on the contract address with this name.

    - 0x237AB82C8a7a3EaD975B0B9Fee13c05d792eeC59
        - This will only track events containing this address for the voting token.

    - null,0x97AF9b7d2bff52b9e0C9C4B437A148A56658e763
        - This will only track events containing this address as the contract address for the delta.

    - 0x237AB82C8a7a3EaD975B0B9Fee13c05d792eeC59,null,ID
        - This will only track events containing this address for the voting token and ID of the delta. Useful only once as the ID will increase with every new delta.

Functions and Events:

- **subscribeEvent**: Create a tracker for blockchain events.

    - **Requirements**: 1 token staked for each subscription.

    - **Event**: SubscribeEvent(address indexed contractAddress, address indexed wallet);
        - Yes, you can track event subscriptions from any wallet.

- **unsubscribeEvent**: Remove a tracker for blockchain events.

  - **Requirements**: Subscription exists.

  - **Event**: UnsubscribeEvent(address indexed contractAddress, address indexed wallet);

Token incentives:

- 1 token must be staked for each subscription created.
- Unsubscribing doesn’t reduce the stake requirement.

APPROVAL

Problem:

- **Hidden**: Token approvals allow smart contracts access to tokens, and the ability to steal them depending on smart contract code. Without a normalized place to track approvals, investors are left vulnerable to theft.

- **Inaccessibility**: Some functions inside smart contracts require token approvals to work like swaps and staking, for instance.

Solution:

- **Tracking**: With this dapp, users can protect themselves from potential attacks by setting token approvals to 0. Also, users can view and adjust token approvals to any number they want as some functions require approval.

- **Access**: Approvals are automatically built into the website for some of Toolbelt’s dapps with user permission required.

Functions and Events:

- **approve**: Gives token approval to a smart contract.

  - **Event**: Approval(address indexed owner, address indexed spender, uint value);
    - Use token address for subscribing to contract.

- **unapprove**: Revokes token approval from a smart contract (sets approval to 0).

AFFILIATE

Problem:

- **Stagnation**: Community growth is ideally a continual trend towards more members. The more avenues towards growth, the stronger the community will expand. Without a program to incentivize the community to expand itself, stagnation is a threat to token price.

Solution:

- **Growth**: By harnessing the potential of the community's marketing abilities, the incentive to earn tokens will expand the community. Competitions will be formed among members to gain the most members and tokens from affiliates. Expanding the community will have a domino effect to transform this small business into a crypto powerhouse helping more users with dapp solutions.

Fee distribution:

- **Portion of transfer fee (for Toolbelt) with affiliate**:
  - 80% to staking rewards.
  - 15% to affiliate.
  - 5% discount to receiver.

- **Portion of affiliate distribution (for other tokens) with affiliate**:
  - 95% to affiliate.
  - 5% to Toolbelt stakers.

Functions and Events:

- **setupAffiliate**: Pairs user wallet to affiliate. Every user transfer will distribute tokens to affiliate.

  - **Requirements**: User is not a smart contract. User cannot affiliate themselves. User allowed only one affiliate.

  - **Event**: SetupAffiliate(address indexed token, address indexed affiliate);

- **affiliateDistribution**: Affiliate earns tokens when user transfers tokens or receives tokens from a smart contract.
    - **Event**: AffiliateDistribution(address indexed token, address indexed affiliate, address indexed wallet, uint weiIn);

LOCKER

Problem:

- **Centralization**: When one or a few people control a token, selfish needs put the token at risk. The potential to destroy a token or change the token's fundamental identity with a click of a button is too much power for a tyrant to wield.

Solution:

- **Decentralization**: With community voting, the token investors decide the changes to the smart contract. This puts the dreams of the many before the dreams of one leader.

Voting:

- **Voting power**: Determined by the number of tokens the voter has, and multiplied by the staking multiplier if tokens are staked in Toolbelt. The stake multiplier can be altered with a vote.

- **Bot resistance**: Contracts are not allowed to vote to prevent flash loan attacks and other botting mechanisms. To prevent bots from spamming wallets for votes, voting power is determined by the number of tokens.

Process/Example:

- **Lock contract**: To enable votes on a contract, the contract owner must relinquish ownership of the contract and give it to the locker. This is done by calling the setupTokenVote function as the owner of a contract. Voting is now allowed for onlyOwner modified functions.

    - **Example**: Let’s say I am the owner of the locker contract and I want my community to vote on changes to the locker contract. I will press this button to setup Toolbelt token as the voting power for changes made to the locker contract.

- **Create delta**: This activates the ability to vote on a function call. Costs an increasing amount of Toolbelt tokens until the timer is reset (avoid calling createDelta for a few days on the contract). The number of days and cost can be altered by a vote.

    - **Example**: I want to change the cost of creating a delta to be cheaper. Since the function is blocked by the onlyOwner modifier, I must initiate a vote. By pressing this button, I initiate a vote to call setWeiBaseFee which changes the weiBaseFee variable from 1 token to 2 tokens.

- **Vote delta**: After a delta (change to a smart contract’s state) is created, users may vote with the token that’s paired to the contract. Votes will be weighted by number of tokens. 75% of users must agree to the delta for a vote to be successful.

    - **Example**: Now that a delta has been created to call the function setWeiBaseFee with the input parameter of 2 tokens (2*10**18 wei). I agree to this change and vote with Toolbelt tokens. Since I am staked, my vote will be worth more.

- **Execute delta**: Anyone can call this function to conclude the process. After the voting period is complete, this will execute the function that was voted on. If the voting percentage didn’t pass 75%, the voted function won’t be called.

    - **Example**: The voting period is complete, and I want to finalize the process. 90% of voters agreed to the change so the function is called and weiBaseFee is set to 2 tokens.

- **Toggle voting**: This initiates a vote in the same structure above to prevent voting or resume voting. By adding an extra barrier of time between creating deltas, bots will have less power to attack and spam function calls.

    - **Example**: People have been spamming deltas and I am tired of voting no for every little change to the smart contract. I want to disallow voting until toggle voting is called again. I press this button to initiate a vote to stop voting. After this delta is executed successfully, delta creation cannot be used until voting is toggled again.

Functions/Events:

- **setupTokenVote**: Used by smart contract owners to give voting power to their token's community.
  - **Requirements**: Only one setup allowed. User must be owner of smart contract.
  - **Event**: SetupToken(address indexed tokenVote, address indexed contractAddress);
- **createDelta**: Initiates a vote to alter the smart contract.
  - **toggleDelta**: Locks voting behind a vote. Same process as createDelta.
  - **Requirements**: Toggle voting is enabled. User has enough Toolbelt tokens.
  - **Event**: CreateDelta(address indexed tokenVote, address indexed contractAddress, uint indexed ID);
- **voteDelta**: Vote on a specific change to the smart contract.
  - **Requirements**: User hasn't voted on this delta yet. User is not a smart contract. User has voting power. Voting period is ongoing.
- **executeDelta**: Finalize the vote by continuing or canceling the function call.
  - **Requirements**: Voting is complete. Function call is successful. Execution can only be called once.
  - **Event**: ExecuteDelta(address indexed tokenVote, address indexed contractAddress, uint indexed ID, bytes encodedOutput);