Gimmer Token Sale Contracts

Lucas Assis Ribeiro

November 22, 2017

1 Gimmer Pre Sale

The Gimmer Pre Sale contract is basically a crowdsale and token contract used as a proxy during the presale for the closed sale before opening to the full public.

1.1 Constructor

1.1.1 Start Time

Start date of the contract in Unix time (see Variables - Start Time)

1.1.2 End Time

End date of the contract in Unix time (see Variables - End Time)

1.1.3 Price

Price for transactions below 3000 ETH (see Variables - Price).

1.1.4 Bonus Price

Price for transactions above 3000 ETH (see Variables - Bonus Price).

1.1.5 Fund Wallet

The wallet that the contract will forward the received Wei to (see Variables - Fund Wallet).

1.1.6 KYC Manager Wallet

The wallet that will start as the KYC Manager wallet (see Variables - KYC Manager Wallet).

1.2 Variables

1.2.1 Start Time

The start date of the contract in Unix time.
24th November 2017 12:00 UTC
Timestamp 1511524800

1.2.2 End Time

The ending date for the contract in Unix time.
2nd January 2018 12:00 UTC
Timestamp 1514894400

1.2.3 Price

The default price to used in the transactions.
11666666 WEI = 1 GMR

1.2.4 Bonus Price

The price to use for transactions that go above the Pre Sale Bonus Min (3000 ETH)
9999999 WEI = 1 GMR

1.2.5 Wallet

The wallet that the Wei funds will be forwarded to

1.2.6 KYC Manager Wallet

The wallet that is able to approve the KYC of supporters. Can be changed by the owner of the contract by calling setKYCManager.

1.2.7 Tokens Sold

The total amount of tokens sold through the Pre Sale contract.

1.2.8 WEI Raised

1.3 Constants

- Pre Sale Token Cap: 15,000,000.00000000 GMR
- Pre Sale Bonus Wei Min: 3,000 ETH
- Pre Sale Wei Min Transaction: 300 ETH (never really used)

1.4 Parameters

- Initial Supply: 0
- Name: "GimmerPreSale"
- Symbol: "GMRP"
- Decimals: 8

1.5 Functions

1.5.1 buyTokens(address beneficiary)

Public

1.5.2 approveUserKYC(address user)

Public

1.5.3 setKYCManager(address newKYCManager)

Public

1.5.4 setKYCManager(address newKYCManager)

Public

1.5.5 userHasKYC(address user)

Public Constant

1.5.6 mint(address to, uint256 amount)

Internal

1.6 Tests

1.6.1 Expected Values

- 30% Price: 11666666 WEI for 0.00000001 GMR
- 40% Price: 9999999 WEI for 0.00000001 GMR

Pricing

- 1 ETH = 780 GMR
- 300 ETH = 234000.01497600 GMR (30% bonus)
- 3001 ETH = 2520840.19158385 GMR (40% bonus)

1.6.2 KYC

- Change KYC Manager to Third account
- Approve Primary account from Third account
- Buy 1 ETH

2 Gimmer Token

Gimmer Token is the contract for the GMR Tokens in the Gimmer ecosystem.
It's a Mintable Token, meaning the Crowdsale actually creates the tokens before each sale.
The token also inherits the Pausable lifecycle contract, as it starts paused - so no trading can occur before the crowd sale is over and the token is unpaused.

2.1 Parameters

- Initial Supply: 0
- Name: "GimmerToken"
- Symbol: "GMR"
- Decimals: 8

3 Gimmer Crowd Sale

Gimmer Crowd Sale is the main contract for the token sale.

3.1 Initial Implementation

Initially the contract inherited straight from OpenZeppelin's Crowdsale, but the end result was more complex than was expected - it works, but with so much redundant data that it should only be used for testing.

The main differences from the OpenZeppelin Crowdsale:

- Our GMR Token has 8 decimals, lower than the 18 used in ETH. The Crowdsale contract uses a rate variable and a multiplier, but in our case we need to divide and truncate.
- The pricing in the contract is dependent on the current date, so either way the pricing variable would just be stored there and unused.

- The contract has a Stage/Phase system, which always keeps tracks of dates and ending times. The crowdsale was also doing that, just adding redundancy/points of failure.
- We have direct access to our GimmerToken interfaced correctly with the contract, so we can call unpause() directly from the contract and eliminate 1 step that is currently needed from the main wallet after the sale (unverified)

3.2 Current Implementation

The current implementation uses the Crowdsale contract, but in a different way; it inherits only from Ownable, and the Crowdsale parts are copy pasted and modified on demand from the OpenZeppelin version.

3.3 Constants

- Pre Sale Token Cap: 15,000,000.00000000 GMR
- Pre Sale Bonus Wei Min: 3,000 ETH
- Pre Sale Wei Min Transaction: 300 ETH
- Token Sale Cap: 100,000,000.00000000 GMR
- Min Token Transaction: 1.00000000 GMR
- Lower Bound Token Limit: 50,000,000.00000000 GMR

3.4 Constructor

3.4.1 Start Date

The date the contract will change from Deployment stage

3.4.2 Sale Token Prices

All the prices the token should have based on each date provided by the following parameter. Current prices being used (unverified):

- 13158496
- 13333332
- 14166666
- 14999999
- 15833332

3.4.3 Sale Token Dates

The ending date of each token pricing phase in Unix time. Current dates being used:

- 2nd January 2018 12:00 UTC or when the 15,000,000 GMR presale cap is reached. Which ever happens first.
- 10th January 11:59 UTC
- 17th January 11:59 UTC
- 24th January 11:59 UTC
- 31st January 11:59 UTC

3.4.4 Sale Wei Limit Without KYC

The limit of Wei an user can spend in this contract before being blocked and needing KYC approval.
5 ETH
This can be updated by the owner of the contract by using updateSaleLimitWithoutKYC, see Functions

3.4.5 Freeze Wallet

The wallet where the frozen funds will go incase we sell lower than the lower bound limit.

3.4.6 Pre Sale Bonus Price

The pricing to be used for the tokens if the user spends more than the min for bonus in PreSale Value being used (for referency only, its wrong):
11263626

3.5 Stages

The contract has 4 distinct stages, not to be confused with the pricing phases (which are basically substages for the PreSale and Sale stages).

3.5.1 Deployment

This is the state the contract starts in. The stage means the contract is waiting for the start date of the crowdsale to move to PreSale. While in Deployment, all buying operations are blocked.
When we get to the starting date, all operations are 'automatically' unlocked. Either the owner of the contract can call forceUpdateState() or the first user to send a transaction to the contract will cause it to update to PreSale stage.

3.5.2 PreSale

The PreSale state is the first stage that allows the sale of tokens, but it's capped. This is considered the first pricing phase (see section 2.7 for Pricing Phases), which will use the pre-discounted price. If the sender buys over the Pre Sale Bonus Wei Min the price drops to the the Bonus Pre Sale Pricing, which has a 40% bonus instead of the 30% PreSale default. While in PreSale time, the contract cannot mint more than the Pre Sale Token Cap (15 million GMR).
Transactions need to be at least 300 ETH to go through, also, if a user buys more than 3000 ETH the transaction will process with the custom presale bonus price.
After the date for the Phase1 to end arrives, we move automatically to the Sale state.
Users need KYC to do any kind of operation on PreSale.

3.5.3 Sale

At the after sale it's where the other 4 token phases happen. No custom pricing rules, just KYC limit and pricing by date.
We begin at Phase2 of the token pricing.
When we reach the end of the last token pricing phase the contract execute the finishContract() function, which:
- Takes 10% of the total amount of tokens sold and forwards them to the creator of the contract
- If we have sold less than the lower bound limit (50 million tokens), after taking the 10% mints the rest of the lower bound limit (so 50 millions - total sold - 10%) and sends them to a frozen wallet.

3.5.4 FinishedSale

This is the stage where the contract is done - it has given ownership of the token back to the owner of the contract

3.6 Pricing Phases

The contract has 5 pricing phases, being the first one the presale phase, and the other 4 standard sale.

3.7 KYC

Due to Know Your Customer rules, the contract keeps track of all accounts that acquire tokens. To buy more than the allowed limit, the user must be flagged by the kycManager account.

3.8 Parameters

- currentStage

- currentTokenPricingPhase
- currentStage

3.9 Functions

- buyTokens
- approveUserKYC
- updateSaleLimitWithoutKYC
- forceUpdateState

3.10 Events

TokenPurchase: