Introduction

GitToken combines the work flows of Git version control system (VCS), GitHub's web-based source code management platform, and the Ethereum network to issue ERC20 tokens, incentivize and reward contributions, and evaluate the fundamentals of projects by integrating token generation with git contributions.

Git is a popular version control system (VCS). GitHub is a web-based source code management (SCM) platform using Git. GitHub is the largest host of source code in the world, managing nearly 57 million repositories and 100 million pull requests for almost 20 million users. (GitHub et. al. 2017)

The emergence of cryptographic networks and assets, such as Ethereum, has created new protocols for sending and managing value. Ethereum is a cryptographic network for running distributed programs; allowing users of Ethereum to send peer-to-peer (P2P) transactions and interact with smart contracts deployed on the network.

Ethereum smart contracts are software applications written in a high-level scripting language and compiled into byte code to be run on a version of the Ethereum Virtual Machine (EVM). The EVM interprets the byte code instruction set and translates the the program into machine code to be executed. (Vitalik et. al. 2017)

GitToken provides a set of open-source software tools and programs to allow any organization using GitHub to issue a token representing a contribution made toward an organizations' git repositories.

Contributions are mapped to GitHub web hook events, and include but are not limited to, creating issues, committing code, merging branches, forking repositories, and reviewing code. GitToken provides a Docker container to deploy a server for configuring and listening to GitHub web hook events.

Contributors receive tokens through interacting with an organization that has configured a GitToken server and has setup a GitHub web hook. When a contributor creates a new event, her/his GitHub login username is provided in the web hook request. Each username is mapped to an Ethereum address in the GitToken contract.

Contributors verify their GitHub identity by authenticating into GitHub using their Open Authorization (OAuth) token credentials. Contributors authenticate themselves with the token contract using the GitToken server authentication URL associated with an organization. For example, the authentication URL for the GitToken GTK token contract is `https://GitToken.org/auth/github`. If a contributor has not yet verified their identity with the contract, their contribution rewards will be held by the contract until their identity has been verified.

The Ethereum ecosystem has adopted a de facto contract interface for transacting value on top of the Ethereum network, the ERC20 protocol. The ERC20 protocol allows tokens to be exchanged over-the-counter (OTC) with private parties using Ethereum contracts. While the standard is still evolving, many developers have used the ERC20 token to represent utility or rights in their projects and have offered tokens to the public to raise funding for open source development. (Aitken 2017)

GitToken will offer GTK tokens to represent contributions made to the organization's GitHub repositories. A portion of tokens issued are automatically auctioned to bidders by the GitToken contract upon each event.

Git Contributions

Incentivizing and rewarding contributions in projects can become fraught with inefficiencies, inequities, and inequalities

GitToken combines the work flows of Git reversion software, GitHub organizations, and Ethereum ERC20 compatible token contracts to better incentivize and reward contributions.

GitToken provides an open-source Express JS web application for listening to GitHub WebHook events triggered by contributions made to an organization repository.

The web application receives post requests made by a configurable GitHub WebHook service, identifies contributors in the request by GitHub login and email, and subsequently issues tokens to the contributor.

GitTokens

Contracts

GitToken.sol

```
pragma solidity ^0.4.11;

import './SafeMath.sol';
import './GitTokenLib.sol';
import './Ownable.sol';

contract GitToken is Ownable {

  using SafeMath for uint;
  using GitTokenLib for GitTokenLib.Data;
  GitTokenLib.Data gittoken;

  event Approval(address indexed owner, address indexed spender, uint valu
e);
  event Transfer(address indexed from, address indexed to, uint value);
  event Contribution(address indexed contributor, uint value, uint date, st
ring rewardType);
  event ContributorVerified(address indexed contributor, uint date);
  /*event ConfigUpdated();*/

  function GitToken(
    address _contributor,
    string _email,
    string _organization,
    string _symbol,
    uint _decimals
  ) {
    if (_contributor != 0x0) { owner[_contributor] = true; }
    gittoken.totalSupply = 0;
    gittoken.organization = _organization;
    gittoken.symbol = _symbol;
    gittoken.decimals = _decimals;
    // Set initial contributor email & address
    gittoken.contributorEmails[msg.sender] = _email;
    gittoken.contributorEmails[_contributor] = _email;
    gittoken.contributorAddresses[_email] = _contributor;

    // Set default rewardValues -- Note, these values are not solidified an
d are untested as to their effectiveness of incentivization;
    // These values are customizable using setRewardValue(uint256 value, st
ring type)

    // Use when setting up the webhook for github
    gittoken.rewardValues['ping']                     = 2500 * 10**_decimal
s;

    gittoken.rewardValues['push']                     = 1000 * 10**_decimal
s;

    // Any time a Commit is commented on.
    gittoken.rewardValues['commitComment']            = 250 * 10**_decimal
s;

     // Any time a Branch or Tag is created.
    gittoken.rewardValues['create']                   = 2500 * 10**_decimal
s;

    // Any time a Branch or Tag is deleted.
    gittoken.rewardValues['delete']                   = 0 * 10**_decimals;

     // Any time a Repository has a new deployment created from the API.
    gittoken.rewardValues['deployment']               = 5000 * 10**_decimal
s;

    // Any time a deployment for a Repository has a status update
    gittoken.rewardValues['deploymentStatus']         = 100 * 10**_decimal
s;

    // Any time a Repository is forked.
    gittoken.rewardValues['fork']                     = 5000 * 10**_decimal
s;

     // Any time a Wiki page is updated.
    gittoken.rewardValues['gollum']                   = 250 * 10**_decimal
s;

    // Any time a GitHub App is installed or uninstalled.
    gittoken.rewardValues['installation']             = 250 * 10**_decimal
s;

    // Any time a repository is added or removed from an organization (? ch
eck this)
    gittoken.rewardValues['installationRepositories'] = 1000 * 10**_decimal
s;

     // Any time a comment on an issue is created, edited, or deleted.
    gittoken.rewardValues['issueComment']             = 250 * 10**_decimal
s;

    // Any time an Issue is assigned, unassigned, labeled, unlabeled, opene
d, edited,
    gittoken.rewardValues['issues']                   = 100 * 10**_decimal
s;

    // Any time a Label is created, edited, or deleted.
    gittoken.rewardValues['label']                    = 100 * 10**_decimal
s;

    // Any time a user purchases, cancels, or changes their GitHub
    gittoken.rewardValues['marketplacePurchase']      = 0 * 10**_decimals;
    // Any time a User is added or removed as a collaborator to a Repositor
y, or has
    gittoken.rewardValues['member']                   = 1000 * 10**_decimal
s;

    // Any time a User is added or removed from a team. Organization hook
s only.
    gittoken.rewardValues['membership']               = 1000 * 10**_decimal
s;

    // Any time a Milestone is created, closed, opened, edited, or deleted.
    gittoken.rewardValues['milestone']                = 15000 * 10**_decima
ls;

    // Any time a user is added, removed, or invited to an Organization.
    gittoken.rewardValues['organization']             = 1000 * 10**_decimal
s;

    // Any time an organization blocks or unblocks a user. Organization hoo
ks only.
    gittoken.rewardValues['orgBlock']                 = 0 * 10**_decimals;

     // Any time a Pages site is built or results in a failed build.
    gittoken.rewardValues['pageBuild']                = 500 * 10**_decimal
s;

    // Any time a Project Card is created, edited, moved, converted to an i
ssue,
    gittoken.rewardValues['projectCard']              = 250 * 10**_decimal
s;

    // Any time a Project Column is created, edited, moved, or deleted.
    gittoken.rewardValues['projectColumn']            = 250 * 10**_decimal
s;

    gittoken.rewardValues['pull_request']             = 1000 * 10**_decimal
s;

  }

  function totalSupply() constant returns (uint) {
    return gittoken.totalSupply;
  }

  function decimals() constant returns (uint) {
    return gittoken.decimals;
  }

  function organization() constant returns (string) {
    return gittoken.organization;
  }
  function symbol() constant returns (string) {
    return gittoken.symbol;
  }
  /*
   * ERC20 Methods
   */
  function transfer(address _to, uint _value) public onlyPayloadSize(2 * 3
2) returns (bool) {
    if(!gittoken._transfer(_to, _value)) {
      throw;
    } else {
      Transfer(msg.sender, _to, _value);
    }
  }

  function balanceOf(address _contributor) constant returns (uint) {
    return gittoken.balances[_contributor];
  }

  function transferFrom(address _from, address _to, uint _value) public onl
yPayloadSize(3 * 32) {
    if(!gittoken._transferFrom(_from, _to, _value)) {
      throw;
    } else {
      Transfer(_from, _to, _value);
    }
  }

  function approve(address _spender, uint _value) public onlyPayloadSize
(2 * 32) {
    // Explicitly check if the approved address already has an allowance,
    // Ensure the approver must reset the approved value to 0 before changi
ng to the desired amount.
    // see: https://github.com/ethereum/EIPs/issues/20#issuecomment-2635247
29
    if(_value > 0 && gittoken.allowed[msg.sender][_spender] > 0) {
      throw;
    } else {
      gittoken.allowed[msg.sender][_spender] = _value;
      Approval(msg.sender, _spender, _value);
    }
  }

  function allowance(address _owner, address _spender) constant returns (ui
nt) {
    return gittoken.allowed[_owner][_spender];
  }


  /**
```

```
   * GitToken Setter (State Changing) Functions
   */
  function setRewardValue(
    uint256 _rewardValue,
    string _rewardType
  ) onlyOwner public returns (bool) {
    gittoken.rewardValues[_rewardType] = _rewardValue;
    return true;
  }

  function verifyContributor(address _contributor, string _email) onlyOwne
r public returns (bool) {
    /*gittoken.emailVerification[_email] = keccak256(_code);
    return true;*/
    if(!gittoken._verifyContributor(_contributor, _email)) {
      throw;
    } else {
      ContributorVerified(_contributor, now);
      return true;
    }

  }

  function setContributor(string _email, bytes _code) public returns (boo
l) {
    if (!gittoken._setContributor(_email, _code)) {
      throw;
    } else {
      return true;
    }
  }

  function rewardContributor(
    string _email,
    string _rewardType,
    uint _rewardBonus
  ) onlyOwner public returns (bool) {
    if(!gittoken._rewardContributor(_email, _rewardType, _rewardBonus)) {
      throw;
    } else {
      address _contributor = gittoken.contributorAddresses[_email];
      uint _value = gittoken.rewardValues[_rewardType].add(_rewardBonus);
      Contribution(_contributor, _value, now, _rewardType);
      return true;
    }
  }


  /**
   * GitToken Getter Functions
   */
  function getRewardDetails(string _rewardType) constant returns (uint25
6) {
    return gittoken.rewardValues[_rewardType];
  }

  function getContributorAddress(string _email) constant returns (addres
s) {
    return gittoken.contributorAddresses[_email];
  }

  function getUnclaimedRewards(string _email) constant returns (uint) {
    return gittoken.unclaimedRewards[_email];
  }

  function getOrganization() constant returns (string) {
    return gittoken.organization;
  }

  /**
   * @dev Fix for the ERC20 short address attack.
   */
  modifier onlyPayloadSize(uint size) {
     if(msg.data.length < size + 4) {
       throw;
     }
     _;
  }


}
```

GitTokenLib.sol

```
pragma solidity ^0.4.11;

import './SafeMath.sol';

library GitTokenLib {

  using SafeMath for uint;

  struct Data {
    uint totalSupply;
    uint decimals;
    string organization;
    string symbol;
    mapping(string => uint256) rewardValues;
    mapping(address => string) contributorEmails;
    mapping(string => address) contributorAddresses;
    mapping(address => mapping(address => uint)) allowed;
    mapping(address => uint) balances;
    mapping(string => uint) unclaimedRewards;
    mapping(string => bytes32) emailVerification;
  }

  /**/
  function _transfer(
    Data storage self,
    address _to,
    uint _value
  ) internal returns (bool) {
    self.balances[msg.sender] = self.balances[msg.sender].sub(_value);
    self.balances[_to] = self.balances[_to].add(_value);
    return true;
  }

  /**/
  function _transferFrom(
    Data storage self,
    address _from,
    address _to,
    uint _value
  ) internal returns (bool) {
    // Check if msg.sender has sufficient allowance;
    // Check is handled by SafeMath library _allowance.sub(_value);
    uint _allowance = self.allowed[_from][msg.sender];
    self.allowed[_from][msg.sender] = _allowance.sub(_value);

    // Update balances
    self.balances[_to] = self.balances[_to].add(_value);
    self.balances[_from] = self.balances[_from].sub(_value);

    return true;
  }

  function _rewardContributor (
    Data storage self,
    string _email,
    string _rewardType,
    uint _rewardBonus
  ) internal returns (bool) {
    uint _value = self.rewardValues[_rewardType].add(_rewardBonus);
    address _contributor = self.contributorAddresses[_email];

    if(_value == 0) {
      throw;
    } else {
      self.totalSupply = self.totalSupply.add(_value);

      if (_contributor == 0x0){
        self.unclaimedRewards[_email] = self.unclaimedRewards[_email].add(_
value);
      } else {
        self.balances[_contributor] = self.balances[_contributor].add(_valu
e);
      }

      return true;
    }
  }

  function _verifyContributor(
    Data storage self,
    address _contributor,
    string _email
  ) internal returns (bool) {
    if (_contributor == 0x0) {
      throw;
    }

    if (self.unclaimedRewards[_email] > 0) {
      // Transfer all previously unclaimed rewards of an email to an addres
s;
      // Add to existing balance in case contributor has multiple emails
      self.balances[_contributor] = self.balances[_contributor].add(self.un
claimedRewards[_email]);
      self.unclaimedRewards[_email] = 0;
    }
    self.contributorEmails[_contributor] = _email;
    self.contributorAddresses[_email] = _contributor;
    return true;
  }
  function _setContributor(
    Data storage self,
    string _email,
    bytes _code
  ) internal returns (bool) {
    if (self.emailVerification[_email] != keccak256(_code)) {
      throw;
    }

    if (self.unclaimedRewards[_email] > 0) {
      // Transfer all previously unclaimed rewards of an email to an addres
s;
      // Add to existing balance in case contributor has multiple emails
      self.balances[msg.sender] = self.balances[msg.sender].add(self.unclai
medRewards[_email]);
      self.unclaimedRewards[_email] = 0;
    }
    self.contributorEmails[msg.sender] = _email;
    self.contributorAddresses[_email] = msg.sender;
    return true;
  }

}
```

Token Auctioneering

Ethereum ERC20 tokens have recently provided a new mechanism for funding projects. While tokens have brought liquidity to start-ups, it has also brought mis-pricing and speculation.

Many of the projects that have offered tokens to build their projects have used Git and GitHub to manage and develop software collaboratively, yet independently.

There is a clear relationship between the number of git contributions a project has and the market capitalization of the project.

Evaluation of Open Source Projects

```
TESTER = document.getElementById('test')
Plotly.plot(TESTER, [{
  x: [38438497236, 18401040701, 1376206596, 235578951, 228864420, 87398581, 56
23927],
  y: [14409, 8680, 8299, 3958, 258, 1492, 706],
  name: 'Contributions',
  text: ['Bitcoin', 'Ethereum', 'Ethereum Classic', 'Golem', 'Gnosis', 'Sta
tus', 'Aragon'],
  textposition: 'top center',
  marker: {
    size: [38.43, 18.40, 1.37, .23, .22, .08, .05]
  },
  mode: 'markers+text'
}],{
  height: 600,
  margin: { t: 100, l: 50 },
  title: 'Git Commits Vs. Market Capitalization'
});
```

GitToken maps the total supply of the token to contributions made to the project.

Authentication

A contributor verifies their identity by associating an Ethereum address to their OAuth GitHub credentials using the web application user interface.

References

Aitken. 2017. “Gnosis' Prediction Market Scores $12.5M in 'Record-Breaking' Crypto Auction.” *Https://Www.forbes.com/Sites/Rogeraitken/2017/04/24/Gnosis-Prediction-Market-Scores-12-5m-in-Record-Breaking-Crypto-Auction/#3afec93ce87d*. Accessed July 11.

GitHub et. al. 2017. “Celebrating Nine Years of GitHub with an Anniversary Sale.” *Https://Github.com/Blog/2345-Celebrating-Nine-Years-of-Github-with-an-Anniversary-Sale*. Accessed July 12.

Vitalik et. al. 2017. “A Next-Generation Smart Contract and Decentralized Application Platform.” *Https://Github.com/Ethereum/Wiki/Wiki/White-Paper*. Accessed July 12.