DefiOS: A Peer-to-Peer mechanism for incentivizing open source development

Priyesh Srivastava [1] and Abhisek Basu [3]

Abstract. A peer-to-peer mechanism for incentivizing open-source development that involves contributors, repository maintainers, and power users is the foundational prerequisite of a more resilient and decentralized internet. Grants by non-profit foundations and patronage by tech giants have been critical to the survival and growth of some projects, but this solution is not scalable and more importantly not in the spirit of open source. To remedy this, we propose a novel ecosystem of tools called DefiOS that allows every repository to become an on-chain DAO with project native tokens. The DAO contract maintains a history of commit hashes on-chain (along with the corresponding working directory hashes) as a proof of contribution that anyone can easily verify. An attacker cannot alter this history without redoing the entire history of commit hashes, starting from the first commit as each commit hash value depends on the previous commit hash value. We ensure verifiability of contributions on-chain by leveraging the merkle trees that git uses to generate hashes and the public-private key dynamics of the blockchain. In this way, DefiOS plans to lay the foundation of a future where open-source and free software rivals closed source and proprietary software, whilst remaining scalable and profitable in the long run.

Keywords: DeFi, Open Source Software, Merkle Trees.

1 Introduction

The internet stands on the back of open-source software. Even a large number of corporate software has begun to rely on the same, from UI libraries to database connectors and backend frameworks. Meanwhile, maintenance of open source code and support for maintainers have been dismal or restricted to non-profit initiatives. The media hype and summer fellowships around open source would lead you into the false comfort that open source projects are lavishly funded. While you could point to projects like Linux, Docker, Redhat, GoLang and React, these are exceptions and not the rule. But there are many reasons why even the Red Hat model fails, the central one being that the business model simply does not enable adequate funding of ongoing investments. As a result, there is minimal product differentiation leading to limited pricing power and corresponding lack of revenue. Pure open source companies have other factors stacked against them as well. Product roadmaps and requirements are often left to a distributed group of unincentivized developers. The complexities of defining and controlling a stable roadmap versus innovating quickly enough to prevent a fork is vicious and complex for small projects.

1.1 Why should we fund open source?

Firstly, most companies use a large number of open source libraries and it is commercially infeasible to divert manpower into the process of forking existing libraries then developing expertise in them and finally monkey patching to get desired behavior. Secondly, as an analysis of recent outages and exploits such as the left D pad attack showcase, our dependency trees are fragile and keeping your fork closed source will require spending much more money on discovering and resolving bugs.

1.2 How does DeFi solve the problem?

The reason why this problem cannot be solved by conventional finance like Venture Capital (VC) is: there is no business model, the project is not structured as a C-corp with formal founders, and most open-source projects cannot possibly return investment at VC scale. Another approach taken by companies such as Google, Facebook, IBM, and Microsoft have taken, can be described as a mixture of acquisitions and grants. However, it is clear that this funding model will also not scale. Firstly, grants are distributed in a very closed source manner to specific individuals and do not benefit the entire community. Secondly, the grants have very little impact on the roadmap of the project because it is not mapped to individual checkpoints in the product's roadmap. However what open source projects have going for them is utility that lies in the present i.e, if they went down or had exploits; some of the biggest names in technology would be impacted severely. In addition, there is a strong community around most projects and using DeFi to exchange value within the community only reinforces the original purpose of the technology.

2 Core Solution

The gist of our solution is to allow developers to convert their open-source repositories into DAOs which issue project native tokens to code contributors. Companies who utilize that project extensively would then be able to purchase these tokens and then stake them behind various GitHub issues (i.e., bug fixes and feature requests) to incentivize developers to prioritize those issues. Over time the value of this token will increase as more and more companies stake these tokens on GitHub issues in the repository. By allowing companies to vote with their wallets for features to be shipped, DefiOS aims to usher in a new economy where all stakeholders have incentives to support, build and promote the ownership of open-source projects. This will ensure that open-source software can consistently rival the best proprietary offerings in performance and usability. You can watch a video summarizing the complete workings of DefiOS with an example project.

Implementing DefiOS required us to answer two very important questions: **how do we ensure that a malicious user does not convert your repository into a DAO** and **how do we ensure on-chain that the rewards have been provided to the right person?** We answer these two questions comprehensively in the following subsections.

2.1 DAO Creation

The ideal solution would've been to start with matching the public key of the individual calling the create_dao function with the public key in the SOL names database corresponding to their github_username.sol, as shown in figure 1.

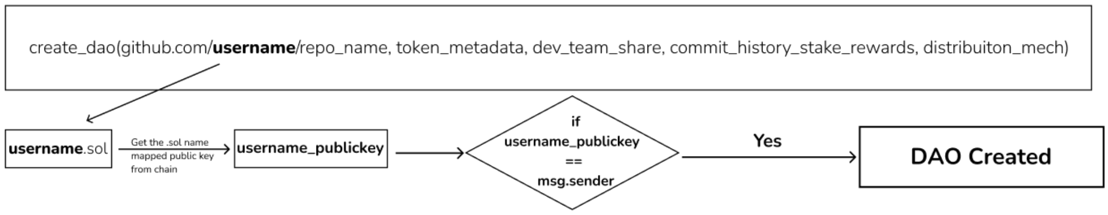
create_dao(github.com/**username**/repo_name, token_metadata, dev_team_share, commit_history_stake_rewards, distribuiton_mech)
username.sol
Get the .sol name mapped public key from chain
username_publickey
if username_publickey == msg.sender
Yes
DAO Created

Fig. 1. DAO creation flowchart where Github username acts as Sol name

However the issue with this solution is that the adoption of SOL names is not widespread enough throughout the broader open source community to guarantee matching GitHub usernames and SOL names. This leaves room for malicious actors to acquire SOL names of prominent GitHub users and create a DAO with that fake identity.

To remedy this, we have created a more practical solution that can be immediately implemented which is based on the premise that only repository owners can alter repository names. We ask the users to set the last 64 characters of their repository names to be the hexadecimal representation of the base58 decoded byte value of their public key. The repository name can then be reverted to its original value once the DAO creation process has been completed. While this method, as shown in figure 2 requires manual intervention on behalf of the repository owner, it is a one time process and **inherits the same level of security as the L1 chain (i.e., Solana)**.

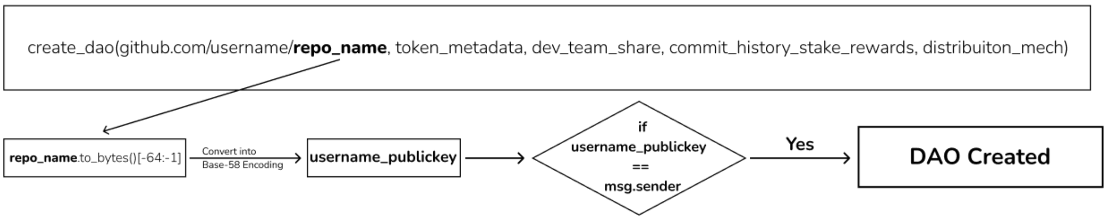
create_dao(github.com/username/repo_name, token_metadata, dev_team_share, commit_history_stake_rewards, distribuiton_mech)
repo_name.to_bytes()[-64:-1]
Convert into
Base-58 Encoding
username_publickey
if
username_publickey
==
msg.sender
Yes
DAO Created

Fig. 2. DAO creation flowchart where repo name ends with public key

2.2 Reward Distribution - Proof of Contribution

To understand how we determine reward distribution on-chain, we first need to investigate what makes each contribution(commit) unique, i.e., the commit hash. Let us look at how each hash gets generated using figure 3. As we see each commit hash depends not only on the hash of the previous commit but also on the SHA1 hash of the working directory of the repository. The hash of the working directory is calculated by recursively hashing all the files in the working directory and combining them together as shown here. We can use this to alter the tree hash by making a commit where the only change in the working directory is adding or removing your public key (i.e., hex of base58 decoded version of public key) as shown here. This will allow the contributor to successfully claim on-chain that the codebase which solves the given issue was written by him. This on-chain record will allow companies to determine the right contributor by using eqns. 1 and 2; thus making reward distribution transparent and fair.

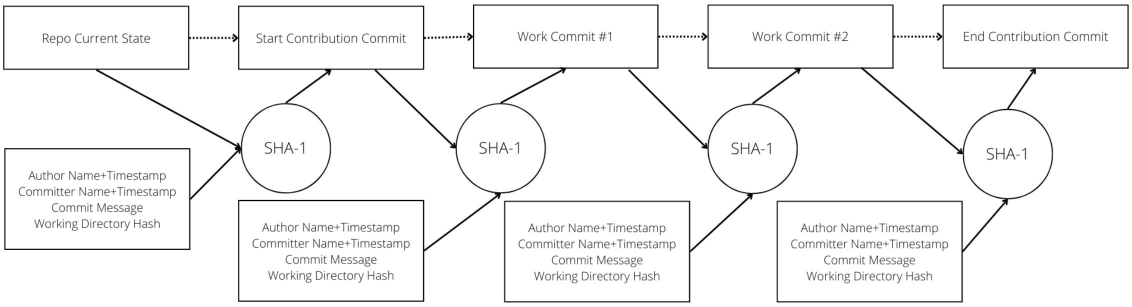

Fig. 3. Commit hash calculations

$$Tree\_Hash_{StartContributionCommit} = SHA1\left(dev\_public\_key + Tree\_Hash_{RepoCurrentState}\right) \tag{1}$$

$$Tree\_Hash_{WorkCommit\#2} = SHA1(dev\_public\_key + Tree\_Hash_{EndContributionCommit}) \tag{2}$$

Using Eq.1 and 2 to determine proof of contribution are very secure because any attacker first needs to reverse engineer which public key produced the change in tree hashes in the commits that indicate the start and end of contribution **i.e. break SHA1**. But even after this, the attacker needs to discover the private key required to sign the relevant public key. Therefore this proof not only inherits the security of the L1 but adds an additional layer of encryption on top of it. A second possible vector of attack is when a malicious actor that tries to create an alternate commit history on chain in order to stake a claim on an solution they did not submit. However, there are 2 reasons why this will fail: **firstly**, commit history hashes form a chain and so it is very easy to dispute invalid commits and **secondly**, when an award is approved, all the people who staked their tokens on the validity of an alternate chain of commit histories lose their staked amounts.

2.3 Tokenomics - DAO Tokens

To fully appreciate how DefiOS ensures fair reward distribution, it is crucial to understand the tokenomics that apply to the project native tokens of all the GitHub repositories that have been converted into DAOs by DefiOS. Let us take an example project called Foobar, whose maintainer has chosen the symbol **FBT** for his project tokens.

Token Supply: 100 million FBT - Fixed
Standard: ERC20/SPL tokens
Utility: Governance, Staking
Staking Rewards: 0.5% of developer rewards for task completion distributed among stakers proportional to the staked amount
DefiOS Fees: 0.25% of developer rewards for task completion

While the governance of DAOs using ERC20/SPL style tokens is widespread and relatively straightforward to understand, let us focus on the second utility of the project native tokens i.e. staking. As we have seen above, the reward distribution mechanism cannot be exploited by malicious actors because the **tree hash** depends on the public key of the contributor. However, we require someone to bring in the history of commit hash and their corresponding tree hashes on-chain. This is where the staking utility of **FBT** comes in. We require everyone to stake a certain amount behind each the validity of each commit hash in the list of commits on-chain.

So now, if a third party observer proves that the chain of commit hashes and their corresponding tree hashes are incorrect, they can upload their version on chain and DefiOS can ensure all the token holders who put in the incorrect data lose their staked amounts. **This ensures an honest commit history on-chain and securing a vital component of the reward distribution**.

3 The DOS token

DOS is the official token of DefiOS, which is backed by a treasury of 1% of each of the project native tokens of all the GitHub repositories that we convert into DAOs. The DOS token being backed by these diverse and creative projects provides an additional reason for why it will continue to accrue value over time. This token will also gain additional utility when the final objectives of our roadmap have been implemented on mainnet.

Total Supply: 100 million DOS [Fixed]
Token Standard: ERC20/SPL tokens
Utility: Building oracles, NFT minting, Index ranking
Holder Incentive: 30% of the fees collected by DefiOS per completed GitHub issues

Having a fixed supply of the DOS token allows us to ensure that inflation doesn't affect the token's value, while the staking of project native tokens allows us to ensure that there is never a barrier of entry for new open-source projects.

4 Roadmap

Our revolutionary solution offers use-cases which expand and complement the core elements of our technology, making it viable and sustainable in the long term. Following are the plans for features which will definitely be implemented, in chronological order, with a rough deadline of 5 months for each item. Each of these features will have the same 5 step lifecyle, consisting of: **prototype, development, closed beta, testnet beta, and production release on mainnet**.

4.1 Web Extension

The most fundamental part of our ecosystem is the chrome web extension, which will work with version control systems like GitHub, GitLab, and Radicle. This will allow developers to convert their open source repositories into DAOs and distribute tokens to existing contributors based on certain criteria. It will also allow companies to create GitHub issues on the extension itself and stake project native tokens behind each issue. This feature is in step 2 of its lifecycle and the final result of step 1 can be found here.

4.2 On-chain Dispute Resolution

One of the differentiating features about our product is how it records proof of contribution on-chain and this allows us to seamlessly integrate with decentralized arbitration services like Kleros, for resolving disputes about remuneration between the feature proposer, repository maintainers, and the developer who delivered that particular feature.

4.3 Demand-side Liquidity Aggregation

Since each project will have its own native token, it is of utmost importance to ensure that consumers (companies) are able to readily convert from one project token to another without breaking the bank on gas fees and impacting user experience. We will accomplish this by deploying the Pool & Vault architecture of BalancerV2 as shown in figure 4. This architecture separates the token accounting and management from price calculations. This separation simplifies DAO token pools, since they no longer need to actively manage their assets. It also drastically reduces gas fees for multi-step transactions since the vault contract holds all tokens.

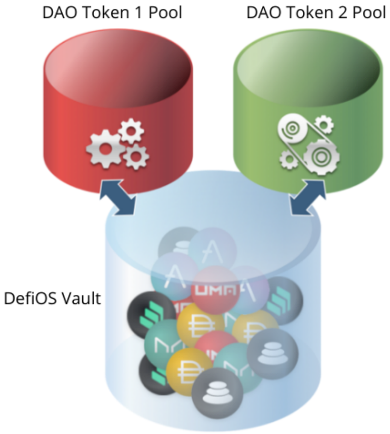

Fig. 4. DefiOS Pool & Vault Architecture

4.4 Governance and Incentives Integration

We are essentially converting open source repositories into DAOs and therefore we need to build DAO management software that allows people that allows the community to frictionlessly handle daily operations while also allowing them to set custom governance rules. We expect the project native tokens to have specific utilities in relevant project ecosystems too, and it remains to be seen the creative and intelligent ways repository owners and the open-source community utilize this flexibility and create value for developers, issue creators, and the community at large.

4.5 Dashboard App - Companies

Therefore, our solution to onboard conventional companies is to provide them with an application which integrates with their JIRA boards and codebase. This seamlessly allows all product and engineering managers to prioritize their budgets based on how important each requested feature is, how important any open-source dependencies are, or if there are any issues in the open-source dependencies they're leveraging that they'd want to prioritize a fix for. There is always an opportunity cost involved in involving a corporation's own engineers in working on underlying open-source blocks which are not in their roadmap. This app will allow them to allocate budgets efficiently and "hire" contributors of those open-source projects with verifiable experience on the repository.

4.6 Taxation Integration - Dashboard App

Established companies must abide by taxation laws in their jurisdiction, and expenses on open source need to be underwritten as either payroll expenses, freelance expenses, or non-profit, depending on their location. So the dashboard app we build for companies will integrate with their tax filing software to export records and/or receipts of their expenditure for appropriate handling and processing by the company's legal personnel.

4.7 Network Indexing & Skill NFTs

One of the most important second order use cases that stem from our core technology is determining a contributor's skill set through their contribution across multiple projects. Since we already have proof of contribution on-chain, it is relatively straightforward to incentivize creation of a range of oracles from off-chain metadata like tech stack, amount of work, variety of work and so on, corresponding to the developer contributions. This oracle data will be further utilized to mint skill NFTs that can be stored in the developer's wallets as the proof of expertise/experience.

4.8 Network Mining

A consequence of wide adoption of our solution would mean that open-source and free software would soon rival the software released by private companies. With each open-source project choosing to be a DAO, it is natural that the inbuilt incentive mechanisms would incentivize developers to keep working on the project, for a community to rally around these open-source alternatives and help support each project, and make it much easier to attract talent and allocate resources towards making open-source projects successful. The network that we'll have as a consequence of this will be leveraged to build a global bounty board, a global launch pad for open-source projects, and a one-stop solution for companies and open-source projects to hire verified engineers based on their relevant skills. This eliminates old models like looking at LinkedIn resumes, setting up meetings and interviews, and facilitates a future where pseudonymous developers can work at their own pace from anywhere and contribute to open-source projects whilst earning a fair income.

5 Moonshots

While DefiOS has an exact and extensive roadmap, our ecosystem will cater to developers and open-source projects. While this market alone is likely to double in 5 years, the general principles of the protocol have scope in different verticals as well. These use cases are labeled as moonshots because the verticals in question lie outside our current team's core competencies and have not yet been the subject of detailed research and exploration.

5.1 Open Source Creators

One of the major reasons why products fail is that they don't spend effectively on market feedback and keep building in silence. Marketers, influencers and designers with a verifiable track record of helping companies succeed will help solve this, and companies can leverage this talent either through bounties through our bounty board, or through job listings in our global task board. In order to expand our current protocol to serve this market, we need to solve only 2 problems: **proof of engagement**, and **equivalent of repository for each type of creator**.