tea.

A Decentralized Protocol for Remunerating the Open Source Ecosystem

Max HowellTimothy Lewis
Thomas Borrel

Abstract

The Internet is in large part composed of numerous different open-source projects and has been since its inception. Every time a new project emerges, proves its worth, and gains users, it becomes a fundamental building block in the tower upon which more software is built. Each time enabling newer and greater innovation than could have been imagined before. The truth is that modern applications are almost entirely open-source with a tiny sliver of proprietary source code—the cherry on top, yet just the cherry all the same. Of all professions, software developers are among the best paid, and yet open-source (typically) is completely unremunerated. We declare our belief that the entirety of modern human endeavor has been stunted by relying on the smallest percentage of its engineers to altruistically dedicate their time to its improvement. It is time we collectively innovate to build a fair and decentralized system of remuneration for the open-source ecosystem that doesn't fundamentally change the nature of what open-source is, or how it is built.

Disclaimer

This white paper is a draft and the information set out herein is of a preliminary nature. Consequently, neither the authors nor any of their respective affiliates assume any responsibility that the information set out herein is final or correct and each of the foregoing disclaims, to the fullest extent permitted by applicable law, any and all liability whether arising in tort, contract or otherwise in respect of this white paper. Neither this white paper nor anything contained herein shall form the basis of or be relied on in connection with or act as an inducement to enter into any contract or commitment whatsoever.

Nothing in this white paper constitutes an offer to sell, or a solicitation to purchase, any tokens discussed herein. In any event, were this white paper to be deemed to be such an offer or solicitation, no such offer or solicitation is intended or conveyed by this white paper in any jurisdiction where it is unlawful to do so, where such an offer or solicitation would require a license or registration, or where such an offer or solicitation is subject to restrictions. In particular, any tokens discussed herein have not been, and, as of the date of issuance of this white paper, are not intended to be, registered under the securities or similar laws of any jurisdiction, whether or not such jurisdiction considers such tokens to be a security or similar instrument and may not be offered or sold in any jurisdiction where to do so would constitute a violation of the relevant laws of such jurisdiction.

Introduction

Freely available, freely maintained, and licensed for liberal modification; the creators and maintainers of open-source are almost always unremunerated. Yet Web 2.0 has accrued fortunes off the backs of this free labor.

Open-source is a labor of love that butts up against the real-world limits of what can be proffered without remuneration. The truthful result of this is genuinely useful projects that don't quite reach their potential. Some projects end up littered with security issues that nobody can afford the time to find and fix, yet black hats and scammers have both the time and incentive. Others get "forked" by enterprises that wrap a business model around the open-source code and generate direct revenue from the work of benevolent developers.

The problem is widely known and attempts have been made toward remunerating open source developers. The most common solutions involve variations on sponsorship or bounty systems. Examples include GitCoin, which provides grants to web3 innovators and open source projects, but requires constant funding. Sponsorship makes it possible for consumers of open source to donate to the projects they favor, however, open-source is a tower of bricks—the lower layers long forgotten but still maintained by dedicated engineers and relied upon by even more developers. Only the projects at the top of the tower are typically known and receive sponsorship. This leads to essential bricks that hold up the whole tower going without donation and favorites receiving more than they need. Recently Log4J was a classic example of this scenario only receiving malicious attention due to a zero-day security vulnerability. Yet, it is still barely funded.

JSCore is the base of every Node.js application that exists. It is downloaded 30 million times *per week*... and yet, it is barely funded too.

Recently several bitcoin core developers resigned citing—among other reasons—*a lack of financial compensation* for their decision.

Bounties allow consumers of projects to propose payment for developers to build certain features, thus only remunerating projects for doing things not necessarily in their best interest. And again only rewarding favorites.

In this paper, we propose a decentralized system for fairly remunerating open source developers from the base to the top of the tower and based on their contributions to the entire open source ecosystem—not merely based on how many stars they have on GitHub.

Components

A software developer building an application for distribution needs four things: a browser, a terminal, an editor, and a package manager. Of these four the package manager is what interests us; it controls the tooling and frameworks a developer needs to construct their product. It is within this layer that we see the potential to change how open-source is remunerated.

The Package Manager

The package manager knows what open-source software an application needs to function, from the top of the tower to its base. Every component and component version that has become essential to the tower over the years is known and recorded. It knows that the top of the tower carefully selects its dependencies (the lower levels in the block tower) and that continues down. The package manager is uniquely placed in the developer tool stack to enable automated and correct distribution of value based on genuine, real-world usage.

Package managers always have a remote "registry" that contains package releases, their versions, and their dependencies.

We propose a decentralized registry, on the blockchain, that can be used to distribute value based on how necessary projects are to the health and wealth of the wider Internet. Value can enter the graph at apex points—apps and essential libraries—and then be distributed to the dependencies of those projects and their dependencies ad nauseam, since the registry knows the entire open-source graph.

Additionally, we believe that the package manager must be augmented with sufficient information for developers to assess whether a package and its author can be trusted. This information may be based on reputation, community kudos, information retrieved from decentralized identity (DID) systems[1], other package managers, or other incentive mechanisms that potentially rely on network participants putting economic value at risk.

The Decentralized Registry

So many wheels so many times reinvented. Every package manager has its own package registry duplicating the same metadata again and again. It's time there

[1] See: https://www.w3.org/TR/did-core/

was a single, comprehensive and definitive registry designed and governed by the communities that depend on it. A decentralized, immutable registry that will resist unavailability and prevent malevolent intent.

The Internet is built on tens of thousands of essential open-source components. It's remarkable that thus far, incidents caused by the yanking of essential open-source infrastructure have been minimal. The most famous was the removal of an NPM left-pad dependency in 2016[2] which cascaded into continuous integration and continuous deployment systems as well as plain local computer developers left high and dry for three days, proving without a doubt that the Internet itself was based on fragile systems of development. Other examples have involved active or intentional participation from the package maintainer themselves to sabotage popular packages (See colors.js and fakers.js[3]), or even bad actors looking to profit by pretending to help maintain packages and corrupting them to steal, for example, Bitcoin private keys (See event-stream[4]).

As we progress further towards a future where a literal currency is fundamentally part of our software, we need guarantees that it hasn't been modified by malicious actors. Unfortunately, the majority of tools that we call package managers cannot guarantee that these packages that are built into the apps and dApps that make up the Internet are the unaltered open-source published by their original authors. Microsoft's GitHub has found that 17% of vulnerabilities in software were planted for malicious purposes[5], with some remaining undetected for extended periods (See Webmin 1.890[6]).

A decentralized registry augmented by a reputation system supported by economic incentives designed to expose bad actors and reward good actors may provide the guarantees developer communities have been looking for.

The Storage System

Open source packages deliver a broad range of functionality, some of which may be restricted, or considered unwanted in certain circumstances. Encryption is a great example of that. Encryption may be used for nefarious purposes (see

---

[2] Source: https://www.theregister.com/2016/03/23/npm_left_pad_chaos/

[3] Source: https://fossa.com/blog/npm-packages-colors-faker-corrupted/

[4] Source: https://github.com/dominictarr/event-stream/issues/116

[5] Source https://www.zdnet.com/article/open-source-software-how-many-bugs-are-hidden-there-on-purpose/

[6] See: Source: https://threatpost.com/backdoor-found-in-utility-for-linux/147581/

Phantom Secure, dismantled by law enforcement agencies in March 2018[7]), or maybe compromised to support law enforcement activities (See Operation Ironside (AFP), Operation Greenlight (Europol), and Operation Trojan Shield (FBI)[8] where the FBI operated an "encrypted" communication platform, ANoM, and convinced criminals to use their "encrypted" phones for secure communication). More importantly, encryption is also used to support individual privacy across the globe.

tea is a decentralized protocol that does not intend to filter or sanction packages based on their functionality. While the tea governance may elect to remove proven malicious packages (see the governance section for details), it is critical for the tea system to connect with multiple storage systems, including decentralized storage systems that allow for secure distribution of software packages and ensure censorship resistance. Package maintainers may be provided the option to select the storage system best suited for their needs to securely store and distribute their packages, including the ability to demonstrate that a package is unaltered and correctly replicated.

Network participants

tea's mission is to empower the open source communities and ensure contributors are supported in their relentless pursuit to improve the tools available to developers. In this document, we will distinguish participants through their contributions. Some may contribute code or verify contributed code. Others may provide economic value to support developers and their reputation.

Package maintainers

Package maintainers are software developers who combine open source software with their creation into a package that provides other developers with the tools they need to fuel and support their creativity.

---

[7]Source: https://www.fbi.gov/news/stories/phantom-secure-takedown-031618

[8] Source: https://www.europol.europa.eu/media-press/newsroom/news/800-criminals-arrested-in-biggest-ever-law-enforcement-operation-against-encrypted-communication

tea assumes that package creators maintain their work. Package maintainers are pillars of the open source communities who need to be empowered and rewarded for their ongoing contributions. Should a package maintainer decide to discontinue its maintenance efforts or operate at a pace that is not aligned with the package user's expectations, a separate developer may be transferred the package's maintenance NFT from the original package maintainer or decide to take on the role of package creator by forking the existing package and submitting a new one which they will maintain moving forward, thus becoming themself both package creator and maintainer.

As such situations present themselves, it is important to provide the developer communities with the right tools to determine which packages are being maintained along with the reputation and quality of work of their past and present maintainers. We've too often seen open source work being tampered with and the efforts of many ruined by bad actors.

Although the work of these bad actors is largely discovered and remediated, it is often not until significant damage has been done and victims have paid a price, whether directly or indirectly, through the unwanted use of their data. An example of this would be the event-stream npm package[9]. Event-stream was downloaded over 1.5 million times *per week* and relied upon by over 1,500 packages when a hacker managed to penetrate the open source project, gain the trust of its original author and modify event-stream to depend on a malicious package that would exfiltrate bitcoin wallet credentials to a third-party server. Although tools may be useful in detecting some of these attacks, they cannot always be relied upon, thus leaving an entire community dependent upon each other's diligence and willingness to share their findings.

We propose to introduce incentives described in the tea token section for the open source communities to report their findings constructively so package maintainers are allowed to address these findings before they are exploited.

Package users

Package users are software developers focused on solving a specific problem and who often look for the tools they need in the open source community to experiment quickly and iterate at very little to no cost.

[9] Source: https://medium.com/intrinsic-blog/compromised-npm-package-event-stream-d47d08605502

Package users benefit from the work of package creators and maintainers. Traditionally, a subset may have chosen to support package maintainers through donations or other forms of remuneration, however, this has rarely been the case, mainly due to sponsorship systems being inadequate for the task with a high number of dependencies and significant friction to even start the process. Of course, experimentation and software building needs to remain unencumbered and free of payment gates. Open source must persist and strive as the foundation of software development, empowering all developers, whether beginners or experts.

tea's purpose is to maintain the core values of open source software while providing a decentralized system to remunerate package maintainers for their work. To accomplish that, tea intends to develop, and incentivize others to develop mechanisms for package users to support package maintainers and benefit from their support through unique usage of the `tea` token, as described in the tea token section.

Package supporters and sponsors

In Web2 and Web3, package supporters have often been called "sponsors". They are organizations or package users who use open-source software to build their commercial products, philanthropists looking to support the ecosystem, or entrepreneurs looking to fund teams for the development of components of a larger system.

tea proposes to extend the communities of package supporters to the entire tea community, whether organizations, developers, users, or tech enthusiasts. tea's goal is to implement decentralized incentive mechanisms for any member of the tea community to contribute to the perpetual sustainability and continuous growth of the open-source communities through unique usages of the `tea` token, as described in the tea token section. Package supporters and sponsors are free to decide which packages or package maintainers to support based on their work, beliefs, or any criteria and metric that would influence their decision. Additionally, the support provided by package supporters and sponsors will flow to each package's direct dependencies, thus trusting implicitly the package maintainer to make good choices about their stack.

Provided that the package maintainer offers such service, a package supporter and sponsor may receive a premium support level NFT in return for their support, thus benefiting from accelerated SLAs or more flexible licensing. Additionally, package supporters and sponsors may decide to support

packages or package maintainers and automatically redirect all, or a percentage of their rewards to incentivize teams to build new open-source software. In other words, packages don't need to exist for tea to start pouring in. Nascent projects can be supported just as well as more mature ones.

tea tasters

As new packages or new versions of existing packages are released, the validity of the work needs to be provably demonstrated so package users can decide whether or not to trust both the package and their maintainers. With the tea protocol, this function is provided by the tea tasters.

tea tasters are typically experienced software developers willing to dedicate some of their time to checking the claims associated with a package (functionality, security, semantic versioning, license accuracy, etc.) and stake both their reputation and economic value to demonstrate the outcome of their research and analysis, and support their claim(s). tea tasters receive rewards for their diligence and efforts. At tea, the action of locking `tea` tokens to support your claim and receive rewards (or penalties) based on the consensus on the validity of your claim is called "steeping your `tea`".

Much like package supporters, tea tasters can influence a package and package maintainer's reputation, however, their impact is greater given their role in the validation of a package's security, functionality, and quality. To support their claims, tea tasters will also need to build their reputation. The quality of their work and the economic value they put at risk as they steep their reviews combined with other external data sources will build each tea taster's reputation, bringing more value to their work. For more details on the mechanisms used to influence a package and package maintainer's reputation, please consult the package reputation section.

Protocol overview

The design of a protocol aimed at rewarding open source contributions is mired with challenges. Open-source software is by definition open to all and can, as a result, be subjected to misattribution, appropriation, or malicious tampering. However, the open-source community has consistently demonstrated its willingness to highlight good actors and expose bad actors. The energy spent

reviewing and commenting on other developers' contributions has historically been strictly voluntary, thus creating no incentive for developers to perform such delicate, time-consuming but crucial tasks, let alone report and defend their findings.

How can developers looking for tools know which package to download and which ones to steer clear of? We believe that adequate rewards for open source contributions cannot succeed without both a reputation system or the ability for members of the community to communicate their findings and support (or dissent) for a package, or the work of a developer.

Each developer operates according to their workflow. Whether GUI or CLI fans, they must all be provided with the tools they need to access and contribute to this reputation system, including simple visual and programmable access to the version and reputation of all dependencies within their packages. A clear understanding of which community members are supporting each package, and to what extent they are supporting each package through steeping will contribute to the reputation of each package. Is the package maintainer steeping their work? Are they putting economic value at risk to communicate how much they stand behind their work? All these data points combined can help inform a reputation system for all community members to facilitate their choice. Looking back at the event-stream package hack, the attack was not conducted through the package itself, but rather via one of its dependencies. Visibility across all layers of dependencies will be key to building a trust system. However, considerations such as computation and transaction ("gas") costs will need to be top of mind as the system is designed and built.

Additionally, since our goal is to reward both Web2 and Web3 developers, the intricacies and specifics of each stack make it so that tracking installations and uninstallations of packages could easily fall victim to one or more bad actor(s) including "buying" installations to artificially inflate numbers. An even worse scenario would be the introduction of a fundamental change to the nature of open-source software by creating unnecessary friction with license keys or other deployment tracking mechanisms. We believe that to provide the broadest technological stack coverage, rewards mustn't rely on the tracking of installations, usage, or uninstallations, but rather on incentive mechanisms that encourage the submission of quality packages and the reporting of nefarious or high-risk packages. Only then will the `tea` token be able to represent the total value of open-source software.

Lastly, many packages rely on common dependencies. For example, lodash has 151,209 dependents[10] while chalk has 78,854 dependents[11] or log4js has 3,343 dependents[12]. As more packages are created using the same dependencies, how do we make sure that incentives are distributed fairly and equitably? How do we make sure that the most utilized dependencies are rewarded without starving new or emerging packages and developers? How do we make sure that the incentive system does not end-up steering developers away from niche languages to centralize them where incentives are better? But also, as developers, how do we identify packages with most dependents and build alternatives - leaner, more efficient, better-coded versions of these packages? At tea, we believe that the lack of incentive has impeded the evolution of open-source software. Supported by the right economic incentives and rewards, more developers will be in a position to build, improve and augment open source software for the betterment of an entire community.

Package submission

The submission of a package requires multiple transactions to occur atomically. Specifically, the submission of a package requires a package creator to:

- Register the package with the decentralized registry,
- Upload the package into the decentralized storage system for both resilience, censorship resistance and ease of distribution,
- [Optionally] contribute to the package's reputation and trustworthiness by steeping `tea` tokens.

Failure of any of the three operations will result in the protocol reverting to its previous state, thus eliminating any evidence of the submissions.

Upon successful submission of a package, the package creator will receive a creator NFT to evidence their work and contribution to open source. The package maintainer may decide to transfer the steeping rewards associated with the creator NFT to a third party, however, the reputation associated with the creation and maintenance of the asset will remain with the package maintainer, thus affecting their reputation over time. As their reputation reaches key milestones, package maintainers may be given access to restricted parts of

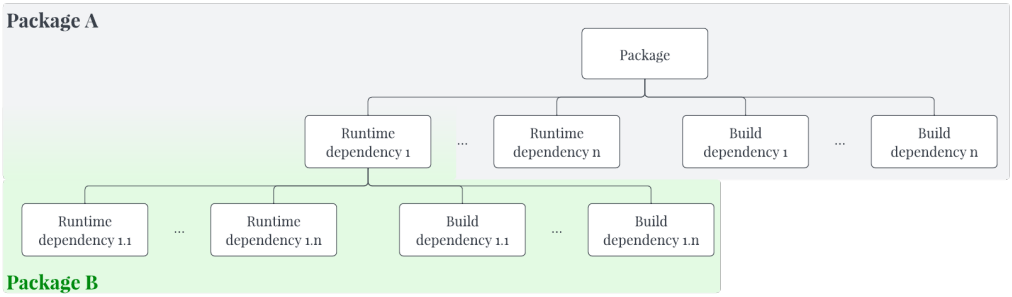
[10] Source: https://www.npmjs.com/package/lodash
[11] Source: https://www.npmjs.com/package/chalk
[12] Source: https://www.npmjs.com/package/log4js

the protocol or accelerated rewards, as decided by the tea governance. For more details on the creator NFT, please consult the creator NFT section.

Dependencies analysis

Package dependencies can run deep as each package often has both dependents and dependencies. To provide a simple methodology that empowers all developers who have contributed open-source software to be rewarded while keeping the creation of the dependencies tree quick and computationally efficient, we propose that only first-level dependencies be verified upon submission of a package.

This design is driven by the hypothesis that each dependency package is itself a package that was submitted to the tea tree. In doing so, each of its dependencies can be mapped and if its dependencies have dependencies themselves, those will be mapped at the time the dependency package is submitted.



The same methodology will be applied for incentive distribution as the steeped tokens are distributed across all dependencies thus recursively steeping the packages listed as dependencies.

Versioning and conflicting dependencies are significant challenges, and troubleshooting them can turn into massive time drains. To address this, we propose each package be subject to a full dependency scan upon submission so we can ensure that the package complies with the following rules for semantic version ranges.

- Packages may only depend on major versions, or greater than specific versions, but for the whole range,
- If a dependency is upgraded to a more recent minor version, the package's minor version must be increased as well,
- If a dependency is upgraded to a more recent major version, the package's major version must be increased as well.

Considering the unnecessary effort imposed upon any user of the package when the above rules are not followed, we propose that a portion of the `tea` token steeped by the package maintainer be burnt to reflect the lack of due diligence on the part of the package maintainer. If a developer forces everyone to juggle their cups, one of them is bound to spill some `tea`. Note that since the dependency scan is expected to occur at the time of submission, no steeping from package supporters and sponsors or tea tasters has happened yet.

Package and package maintainer reputation

Package maintainers may (and should) contribute to their package's reputation and trustworthiness by steeping `tea` tokens. However, a reputation system that relies solely on the author's economic contribution does not provide sufficient user protection and can be subject to Sybil attacks. There are, however, several methodologies that could be applied to prevent Sybil attacks, some of which are described by Nitish Balachandran and Sugata Sanyal in "*A Review of Techniques to Mitigate Sybil Attacks*"[13]. tea is a decentralized protocol, so any use of a trust certification system which relies on a centralized certificate issuance authority would be contrary to its very core. We propose to focus on decentralized approaches to Sybil attack mitigation, and more specifically on methodologies that rely on a large group of network participants who are incentivized to assess and publicly represent the reputation of each package and its creator. We believe that this approach would be congruent with open source and therefore provide a more fertile substrate to foster adoption and trust, and ultimately facilitate the growth of tea.

Similar to the production of blocks on a proof-of-stake blockchain, where non-producing nodes can validate the work of others, and, should that be necessary, highlight a violation of the rules of the network thus leading to a penalization of the bad actor through slashing (destruction of a portion of their stake), we propose a system whereby third-parties (aka tea tasters) would be able to review packages produced by package maintainers and be economically incentivized to behave in the best interest of the open-source software community and its users as well as recognize good behavior and penalize bad behavior. It is critical that this system be both Sybil resistant and

[13] Source: https://arxiv.org/abs/1207.2617

prevent large token holders from materially influencing the protocol or the reputation of specific packages.

Package review by third parties

The review of packages by third parties is an important component of reputation building. A common threat to all reputation-building systems is Sybil attacks, situations whereby a single individual creates multiple representations of themselves for the purpose of leaving a large volume of positive reviews on their own work, thus tricking users into believing their work was reviewed and approved by many.

Blockchain technology, and more specifically staking, offers a unique opportunity for tea to tackle this challenge. Although wallet addresses may be available in infinite quantities, this is not the case with `tea` tokens, whose initial supply is expected to be 10 billion. Additionally, each action performed by developers, from submitting packages to verifying packages or steeping them, will contribute to their reputation, and thus the creation of a unique profile each developer can use to both contribute to the tea community and participate in tea's governance.

By requiring third-party reviewers to steep `tea` tokens and incur the risk of losing a portion of their steeped tokens should they turn out to behave against the interest of the network or be a bad actor, third parties can provide additional credence to a package, and receive a reward, in the form of newly minted `tea` tokens.

We also propose extending the reputation system to the third parties who perform the independent verification of packages - the tea tasters. The completion of a positive review will require two operations to occur atomically:

- the submission of the code review, signed by the tea taster's private key and publicly accessible to all members of the community, along with
- the act of steeping "for" the package (vs. "against" the package), to substantiate their review.

The completion of a negative review that includes one or more critical vulnerabilities will require the tea tasters to contact the package maintainer using a messaging protocol to notify them of the vulnerability and allow them to address the issue. To disincentivize wasting developers' time, such communication will require the tea tasters to steep `tea` tokens.

Upon successful completion of both operations, the tea tasters will receive an NFT as evidence of their work on the specific package and package version. The accumulation of NFTs combined with the steeping ratio of each of the packages reviewed will inform a tea tasters' reputation. As their reputation reaches key milestones, tea tasters may earn access to restricted parts of the protocol or accelerated rewards, as decided by the tea governance.

Outdated or corrupted packages

tea's mission is to reward contributors and participants in the open-source communities, however, rewards are meant to be commensurate with the efforts deployed by package maintainers and tea tasters. Under-maintained, outdated, or corrupted packages are clear indications of package maintainers not living up to the community's expectations or delivering on the trust and support impressed upon them through the steeping of packages. Another manifestation of outdated packages may be the continued use of a legacy language or legacy version of multi-version languages. Packages remaining outdated or corrupt for too long is also an indication that tea tasters need to review package maintainers' work regularly and consistently.

tea tasters are critical members of the open-source communities in that their reviews and associated claims can steer package users towards or away from packages. To ensure that reviews can be trusted on an ongoing basis, we propose a mechanism whereby outdated or corrupted packages may see a portion of their steeped tokens sent to the tea tasters who were first to recognize the lack of maintenance of any package.

Any negative review which outlines a flaw such as a zero-day vulnerability or the use of an outdated dependency and remains open past a grace period defined by governance should be considered a failure on the part of the package maintainer. They have not completed the task they were entrusted with and rewarded for. And neither did the package supporters and sponsors who staked their reputation on the work of the package maintainer, received rewards for it, but yet elected to continue to support the package despite a lack of maintenance.

As packages gain in popularity and usage, with more applications and potentially mission-critical systems depending on them, it is vital that we incentivize developers to discreetly report flaws to the package maintainer and package maintainers to address such flaws before they can be exploited in the wild. Consequently, we propose that any outdated or corrupted package which is subject to one or more substantiated negative reviews and remains in such state

past the governance-defined grace period, see a portion of its steeped tokens be burnt regardless of their origin (package maintainer, package supporters and sponsors or prior tea tasters), while another portion is sent to the tea tasters who submitted the negative reviews. Distribution to all tea tasters could be based on the age of their review and the number of `tea` tokens they steeped for their review.

Creator NFT

Upon successful submission of a package, the package creator will receive a non-fungible token (NFT) to evidence their work and contribution. The holder of this NFT will automatically be the recipient of all rewards associated with the package. Package creators may choose to transfer maintenance ownership over a package to another package maintainer by simply transferring the package's NFT. Upon successful transfer of the NFT, the new owner will automatically become the recipient of future package rewards.

An important part of reputation building relies on the frequency and quantity of quality package submissions. The NFT delivered to package creators as evidence of their work may be used by the reputation system to update a package maintainer's reputation and potentially give them access to parts of the protocol which are reserved for the most active contributors in the open-source communities. However, to prevent new and unwanted attack vectors, such as community members buying their reputation, the transfer of the NFT used to evidence work and contribution will not result in a transfer of reputation. Reputation must remain directly associated with a specific developer's work and must not be transferable.

tea token

Securing the network

tea's desire for sustainability spans beyond the open-source ecosystem itself, to include the environment that surrounds us. While a blockchain may appear as an effective and secure infrastructure to support tea's objectives, we believe that careful consideration must be given to the technology stack upon which the tea system is built.

Scalability, cost-effectiveness, and third-party extensibility are important design considerations that are currently leading us towards building a proof of stake blockchain. In proof of stake, node operators and network participants stake economic value, in the form of the chain's native token, to increase the security of the system. Node operators and network participants receive rewards for the successful production of blocks that comply with the rules of the network and include accurate transaction information. Inactivity (aka node down) or malicious/incorrect activity are penalized through the destruction of a fraction of the staked tokens (aka "*slashing*" event).

The use of a proof of stake blockchain powered by the `tea` token will not only provide `tea` token holders with the opportunity to contribute to the security of the system, by staking `tea` but also help support open-source developers, by steeping `tea`. Economic factors may prevent some developers from staking or steeping `tea`, as such, staking and steeping will be available for a `leaf`, the smallest denomination of `tea` which represents one one hundred millionth (10^{-8}) of a `tea`.

Both applications of the `tea` token serve vital functions in the support and growth of the open-source ecosystem. Staking `tea` will ensure that the tea system continues to operate securely, so all network participants can submit and access packages for the purpose of reviewing them, integrating them into their application, etc, while the steeping of `tea` will support our goals of providing the tools for all network participants to support and use packages that meet quality and dependability requirements, as formulated by the tea community through their support and dissent of each individual package. Care will be taken when defining and implementing staking and steeping parameters, so one does not become parasitic on the other.

Incentives and penalties

As discussed in earlier parts of this document, there can be strong incentives for bad actors to compromise open-source software. With parts of our critical infrastructure running on open source, the honeypots are there and the race to find exploits and other vulnerabilities is on. At tea, we believe that package maintainers are not the ones that should be blamed (although they often are), incentives are. And this is how we propose the `tea` token will solve this issue.

A fair and equitable incentive distribution is absolutely essential to the success of tea. A package like lodash with over 151k dependents is clearly a pillar of open source development and its maintainer deserves to be rewarded as such.

However, a reward system built solely on the number of dependents would prevent innovators from disrupting these monopolies, unless they are sufficiently funded by third parties, or have already accumulated enough resources to self-fund. Such an approach would likely yield a shrinking number of contributors, thus creating the polar opposite of what tea is about.

tea's goal is to represent the value of open-source software and, in doing so, foster its growth by empowering its participants with the resources they need to pursue their passion unencumbered. To accomplish this goal, the tea incentive distribution system needs to carefully consider the steeping ratio of each package and adjust each package's incentive accordingly. As a package steep increases and approaches a governance-defined optimum steeping ratio, its steeping rewards will decrease progressively. When a package exceeds its optimum steeping ratio, steeping rewards will decrease sharply to de-incentivize package supporters and tea tasters from further steeping these packages. This design will create an opportunity for lesser steeped packages (who have a higher associated steeping reward ratio) to become more attractive to both package supporters and tea tasters, and for experienced developers to build alternatives to highly steeped packages, thus creating an opportunity for steeped tokens to be rebalanced. In its initial design, the steeping ratio will be calculated using the circulating supply. This design may be altered by the team community to further improve the system's scalability.

Just like good actors need to be rewarded, bad actors need to be identified and penalized. Open-source software provides many opportunities for bad actors to create pain points and reputational risks for an entire community of developers. From the misappropriation of work to the alteration and redistribution of software packages, or the injection of nefarious code, the war between good and bad actors rages on, often with well-funded bad actors who see the contamination of open source packages as an opportunity to generate more money.

To date, the downside has been relatively minimal with packages potentially banned from digital shelves or subjected to a poor reputation. To address this situation, we propose the introduction of a slashing mechanism. The purpose of this slashing mechanism is to introduce a more material downside that directly affects bad actors' bottom line: economic value. As tea tasters evaluate and analyze the code in newly submitted packages, we propose that tea tasters be provided the tools and incentives to pinpoint and highlight nefarious code so that package users can be made aware of the risks and package

maintainers, package supporters, and sponsors are penalized for submitting and/or supporting nefarious code. To that extent, for all substantiated negative reviews performed per the rules of the network and which have been addressed by the package maintainer within the governance-defined period, the package maintainer should not incur any penalty contrary to the package supporters and sponsors or the tea tasters who provided a positive review of the package in question. For negative reviews performed per the roles of the network which have not been addressed by the package maintainer within the governance-defined period, a fraction of the tokens steeped by the package maintainer, the package supporters and sponsors, and previous tea tasters will be destroyed ("slashed"). Another fraction will be locked into an insurance pool, while the remaining steeped tokens will be distributed across all tea tasters who contributed to the negative review and steeped against the package, based on the number of `tea` tokens they steeped "*against*" the package and for how long their tokens have steeped. In other words, the sooner one or more tea tasters identify and report the flaw according to the rules of the network, the higher the reward they will get for supporting safe and productive open source development.

To prevent community members from randomly voting "*against*" highly steeped packages so that they receive the majority of the penalty, all `tea` tokens steeped "*against*" will not be rewarded with inflation and may be subject to a decay mechanism, thus reducing their value over time.

Governance

Development, sustainability, and adoption of any distributed system cannot be accomplished without proper governance.

We propose that tea includes on-chain governance for critical parameters including (but not limited to) inflation, transaction fees, staking and steeping rewards, or optimum steeping ratio. This functionality will ensure members of the tea community are given the tools to optimize critical parameters over time. To facilitate adoption and a progressive decentralization of the tea system, governance will launch with a simple structure and progressively expand as the tea system continues to mature.

To reduce the attack surface represented by governance and while governance may be upgradeable, at the time of launch, not all parameters associated with the tea system will be subject to governance or support being changed at high frequency. A progressive transition of parameters to open, decentralized governance will ensure the stability and predictability of the

system. As such, we propose that only a predetermined number of parameters be subject to governance votes, and the maximum frequency of their updates be subject to governance votes as well.

Third-party extensibility

As we provide the tools to ignite the long-overdue support of the open-source communities, we believe that part of our mission is to ensure that the tools can be extended by third parties. In addition to providing the infrastructure for developers to build extensions to the protocol, including new ways to innovate and further the support of open source developers, our plans include the potential for other package managers to contribute to the protocol. The dreams and efforts of open source developers have built the innovation that supports our everyday life and we look forward to discovering new uses and extensions for tea. Get your teapots out, shakers ready, and innovate!

Future work and potential community effort

As the tea system continues to mature, we foresee the community deciding and contributing to alterations and extensions through governance. Below are some ideas that we believe may spark some thoughts.

tea wholesalers

The open-source software communities are vibrant and constantly looking to innovate and deliver value. This dedication and altruism lead to the constant building of new software and packages, each one pulling dependencies. As a result, we anticipate the dependencies map to evolve constantly, leading to frequent changes to the staking ratio and therefore rewards. In the future, the tea community may propose the development of a system designed to dynamically monitor the staking ratio for each package and re-balance package supporters' stakes based on their own criteria.

Royalties on package transfer

We recognize that package maintainers may decide to transfer their steeping rewards stream to one or more developers. The governance of such transfer must remain the decision of the package maintainer and their partners, with no interference from tea. Tools will need to be provided for such transfer to be total or partial (perhaps through only a portion of the steeping rewards being redirected to one or more developers, while the remaining rewards continue to flow to the original package maintainer), and for the steeping rewards to flow through a single account controlled by a single network participant, multiple network participants, or automatically distributed across multiple accounts using static or dynamic ratios.

Rewards distribution across multiple maintainers

The maintenance of a package can rely on the work of one more team of developers. Before steeping rewards start to flow, the teams should consider automating the distribution of steeping rewards amongst themselves. How the distribution occurs must be decided by the maintainers themselves, as only they know who contributed and how they should be rewarded.

To accomplish that, each team (or teams) could set up their own decentralized autonomous organization (DAO) and either automate the distribution of rewards, or deploy more complex systems to determine the adequate rewards distribution based on external factors such as a vote from all DAO members, or time-based distributions based on continuous contribution, successful completion of bounties, etc.

Handling of package “forks”

We believe that forks are incredibly important and largely under-utilized. Forks can be an effective tool to develop packages that will compete in functionality, performance, security, and even attention. Forks, as useful as they may be, must recognize the original efforts. Through future work or potential contributions from the tea community, the tea system may be enhanced to require forks to be declared, perhaps even detected at the time a package is submitted. Undeclared forks revealed by tea tasters may result in a portion of the steeped tokens being burnt, transferred to the original package maintainer, and sent to the tea tasters who revealed the fork.

Build dependencies vs. runtime dependencies

At launch, tea may not distinguish build dependencies from runtime dependencies when it comes to steeping rewards distribution, however, provided the tea community feels strongly about making such distinction, enhancements to the steeping rewards distribution algorithm to account for the criticality of each dependency and their contribution to the existence and value of the packages that depend upon them may be proposed by the tea community, voted upon and implemented, should the community decision require it.

Acknowledgments

This white paper would not exist without the support and dedication of many teaophiles . The authors would like to acknowledge Josh Kruger and Jadid Khan for their contribution to the tokenomics, and xxxx for their feedback on the contents of this document.

Glossary of terms

Term	**Definition**
Leaf	Smallest denomination of the tea token. A leaf corresponds to one one hundred millionth (10^{-8}) of a `tea`.
Staking	Action of locking `tea` tokens for the purpose of securing the proof-of-stake network upon which the tea system is built.
Steeping	Action of locking `tea` tokens for the purpose of supporting your claim and receiving rewards (or penalties) based on the general consensus on the validity of your claim.

References

1. https://www.w3.org/TR/did-core/

2. https://www.theregister.com/2016/03/23/npm_left_pad_chaos/
3. https://fossa.com/blog/npm-packages-colors-faker-corrupted/
4. https://github.com/dominictarr/event-stream/issues/116
5. https://www.zdnet.com/article/open-source-software-how-many-bugs-are-hidden-there-on-purpose/
6. https://threatpost.com/backdoor-found-in-utility-for-linux/147581/
7. https://www.fbi.gov/news/stories/phantom-secure-takedown-031618
8. https://www.europol.europa.eu/media-press/newsroom/news/800-criminals-arrested-in-biggest-ever-law-enforcement-operation-against-encrypted-communication
9. https://medium.com/intrinsic-blog/compromised-npm-package-event-stream-d47d08605502
10. https://www.npmjs.com/package/lodash
11. https://www.npmjs.com/package/chalk
12. https://www.npmjs.com/package/log4js
13. https://arxiv.org/abs/1207.2617