TrustSECO: A Distributed Infrastructure for Providing Trust in the Software Ecosystem

Fang Hou[1][0000−0002−8042−3278], Siamak Farshidi[2][0000−0001−6139−921X], and Slinger Jansen[1,3][0000−0003−3752−2868]

[1] Utrecht University, Utrecht, the Netherlands
{h.fang,slinger.jansen}@uu.nl
[2] University of Amsterdam, Amsterdam, the Netherlands
s.farshidi@uva.nl
[3] Lappeenranta University of Technoly, Lappeenranta, Finland

Abstract. The software ecosystem is a trust-rich part of the world. Collaboratively, software engineers trust major hubs in the ecosystem, such as package managers, repository services, and programming language ecosystems. However, trust entails the assumption of risks. In this paper, we lay out the risks we are taking by blindly trusting these hubs when using information systems. Secondly, we present a vision for a trust-recording mechanism in the software ecosystem that mitigates the presented risks. This vision is realized in TrustSECO: a distributed infrastructure that collects, stores, and discloses trust facts about information systems. If our community manages to implement this mechanism, we can create an urgently needed healthy and secure software ecosystem. Finally, we report on the current status of the project.

Keywords: software ecosystems · distributed ledger · software trust · repository mining · software security.

1 Introduction

A software ecosystem (SECO) is a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them [8]. Society is entirely dependent on a healthy worldwide SECO, as every aspect of our society is dependent on information systems and other software. In addition, many worldwide actors rely on the different links in the software supply chain, which works on the basis of trust, hence, trust plays a key role in our society as well as in information systems. We identify software trust as a willingness of the information systems end-users to take risks based on a subjective belief that information system providers will exhibit reliable behavior to provide its required functionalities, operate reliably and consistently without failures, even under uncertainty. Hence the system characteristics, such as reliability, fault-tolerance, safety, or security should be considered as the factors to judge an information system's trustworthiness [2, 14].

Although the worldwide SECO is a trustful environment, it presents many dangers to society. First, as information systems are malleable, they evolve constantly and are evolved by actors that may have bad intentions. Also, most information systems are used for critical infrastructure on which our society depends. In addition, when information systems end-users select a software package, they give a large dosage of trust to the package manager and language ecosystem that they are part of, most of them insufficiently think about security and trust [9].

Thus, attackers just exploit this kind of trust to carry out attacks. For instance, a package registry can be compromised through *Registry Exploitation*: hijacking the account of a registry maintainer or package maintainer, hacking the registry itself, or hacking the registry infrastructure. Furthermore, it is possible to *publish a package with a similar name (a.k.a. Typosquatting) or exactly the same in another registry*, leading to downloading compromised or vulnerable packages. Also, a hacker can *transfer ownership* of an abandoned package to herself and then compromise it. There are also possibilities for open source developers to compromise packages, such as by a disgruntled insider or a malicious contributor. Attackers will use these attack vectors to insert malicious code, infect packages with viruses, provide back doors, and steal sensitive data. These attacks frequently make headlines, and the outcomes are devastating. With the constant increase in the number of detected vulnerabilities, it is time to radically rethink the worldwide SECO, information systems, and the trust we put in them.

In this paper, we present TrustSECO[4], a community-managed infrastructure that underpins the SECO with a trust layer. The infrastructure gathers data on trust in particular software packages and projects. Usually, this data is made available to the SECO hubs, such as package managers and repository websites. With this data, the software end-users can collectively determine whether packages and package versions are reliable, containing vulnerabilities, and are trusted by other users.

The rest of this paper is structured as follows. In Section 2, we outline the key components of TrustSECO: the distributed ledger technology and trust score calculation component. Section 3 describes the trust score calculation, which uses facts from the SECO about information systems to provide trust data to end-users. Section 4 shows the initial vision of the distributed ledger design, which includes ledger design, data sources, and ledger sustainability. Furthermore, we discuss the evaluation of the TrustSECO platform in Section 5. Finally, in Section 6 we summarize our work and provide a status update of the project.

2 TrustSECO Infrastructure

The TrustSECO infrastructure provides a distributed system that enables software end-users to evaluate and install the software based on its trustworthiness, intending to provide a safer SECO. The main object of interest for TrustSECO

[4] `https://secureseco.org/secureseco-introduction/trustseco/`

are versions of software packages, i.e., collections of components that form a coherent whole that can be deployed on a system to provide a set of features to a software end-users.

The main stakeholders are software end-users who install and use the software, and software providers including software producing organizations, organizations that create package managers, and software engineers who create the software or packages. These different stakeholders collect trust data about software packages, by, for instance, looking at past performance, reproducible builds, and vulnerability databases. Furthermore, the stakeholders collect trust data about software providers who have contributed to the software package. In addition, the stakeholders collect trust data about the package manager offering software packages.

To ensure that the community can evenly contribute and make use of the infrastructure, we envision it as a fully distributed system, in particular a Distributed Ledger (DL) containing trust facts. The DL collects trust facts and observations from different stakeholders who participate in the network. Trust facts can be collected from various data sources, for instance, they can be from the participants in the TrustSECO DL, or different data sets in the networks. Subsequently, ecosystem hubs can reuse the data from the TrustSECO DL, base on our trust score calculation mechanism, to establish whether a particular version of a package can be trusted.

An example of how TrustSECO works is illustrated by an example with the Node Package Manager(*npm*). When a software engineer is developing software and including existing *npm* components, she might download them through *npm*. If *npm* was integrated with TrustSECO, TrustSECO can be used to provide warnings about particular configurations. TrustSECO, before downloading and installing a package, can for instance automatically advise the software engineer on the trust rating of the package. If the package (version) is below a particular rating, for instance, set by the software engineer's organization, *npm* can use TrustSECO to select another version of the package or even a wholly different package. Furthermore, TrustSECO can warn the software engineer, when any new vulnerabilities arise in one of the installed packages. In some scenarios, TrustSECO can be used to automatically re-configure a system to ensure that the configuration matches the organization's security requirements.

The TrustSECO infrastructure consists of the following parts: the trust fact DL, the trust fact observer client, the trust score calculation mechanism, and the SECO alert service. We provide an overview of the TrustSECO infrastructure in figure 1.

Distributed Ledger Technology

Distributed Ledger Technology (DLT) has been most successfully applied in supply chain management, which is not surprising, as DLT concerns the interaction between multiple untrusted parties that collaboratively achieve a goal [6].

Hence, we bring DLT into the worldwide SECO, i.e., the complex network of software providers that collaboratively provide software for every computer on earth to make the SECO more secure, reliable, and trustworthy. There are

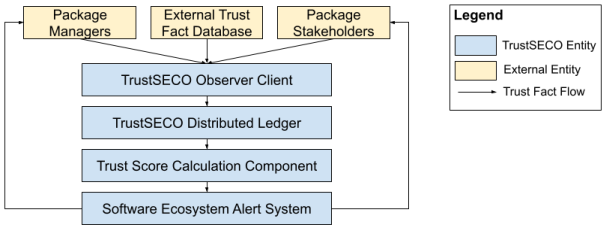

Fig. 1. *The main four components of the TrustSECO infrastructure are visualized. All trust data are stored in the TrustSECO DL. Using the score calculation, the alert system notifies the stakeholders about the status in the SECO.*

two reasons to choose a distributed ledger. First, as the community needs to rely on a trustworthy source, we aim for the trust data to be collected and shared through a consensus mechanism, where the perception of the package will become more reliable and complete as more participants add data about the package. Secondly, as the TrustSECO platform needs to outlive the duration of this academic project, we aim for the TrustSECO platform to be maintained by the community.

The DL we use includes the trust data storage structure and serves as the basis for the TrustSECO implementation. The trust data will be crawled and confirmed by the TrustSECO nodes, i.e., trusted nodes in the network that collaboratively confirm trust facts. Moreover, with the employment of the consensus mechanism, all trust facts will be validated by the nodes. Subsequently, those confirmed the trust data will be compiled into stakeholders' specific trust score calculation and reports about software packages and versions that are used by hubs in the SECO that need trust data to ensure trustful software usage. For a full discussion on the DL and its design, see Section 4.

Trust fact observer client

Trust fact observer client will be executed by the participants in the TrustSECO DL as a mechanism for adding new trust facts to the ledger. Meanwhile it continually monitors the SECO along with package managers for new trust facts and occasionally performs jobs, such as trying to reproduce a build. When these facts are submitted, a set of items are checked, such as the submitter's trustworthiness and observation. An overview of the data stored is found in Table 1. The data must be retrievable relatively fast, even though these data are extensive.

Trust Score Calculation Component

We develop a score calculation mechanism to provide insight into the trustworthiness of the package configuration. This is especially suitable for package managers with self-contained package configurations. The score calculation can be used to ask for particular configurations, such as *"a configuration that provides Python3 and qtbitcointrader with a trust level of at least level X"*. The Configuration Trust Level Calculation can also provide a configuration where X is the highest possible. The configuration algorithm will mostly consider the weakest link principle, i.e., the package with the lowest trust level is the primary determinant for the configuration's combined trust rating. We create a tool that

		Source Created	Source Built	Source Tested	Source Committed to Repo	Package Built	Package Tested	Package Released	Package Published	Package Downloaded	Package Installed	Package Run	Package Removed
Software Engineer													
	Identity?	W	W	W	R/W	R	W	W	R/W	R	R/W	R/W	W
Internal Integrity													
	Syntactically correct	W	W	-	R	R	-	R	R	R	R	R	-
	Style correct	W	W	-	R	R	-	R	R	R	R	R	-
	Reproducible build	-	W	-	R	W	-	R	R	R	R	R	-
	Compiler trusted	-	W	-	R	W	-	R	R	R	R	R	-
	Package integrity	-	-	-	-	W	W	R	R	R	R	R	-
	Code complete	-	-	-	-	-	-	W	R	R	R	R	-
External Integrity													
	Virus scanned	-	-	-	W	W	R	W	R	R	R	R	-
	CVE free	-	-	-	W	W	W	W	W	R	R	R	-
	Most recent version	-	-	-	-	-	-	W	R	R	R	R	-
	Dependency tree up to date	-	W	W	-	W	W	W	W	R	R/W	R	-
Tested													
	Code tested	-	-	W	R	-	W	R	R	R	R	R	-
	Acceptance tested	-	-	-	-	-	W	R	R	R	R	R	-
License													
	License present	-	-	-	-	-	-	W	R	R	R	R	R
	License compliant	W	-	-	-	-	-	R/W	R	R	R	R	R
End User													
	Identity?	-	-	-	-	-	-	-	-	W	R/W	R/W	R/W

Checked on distributed ledger R
Added to distributed ledger W

Table 1. *Each column represents a step in the software development life cycle. Each row indicates an example of a trust fact that we can read or write to the DL, with the goal of securing the worldwide SECO. For convenience, we have shaded the "writes" red, the "reads" green, and the "reads/writes" yellow. Please note that the rows in this table are an incomplete list of the technical factors that we describe further in Section 3.*

supports package managers in finding the most trusted configuration of components and automatically developing a migration path. It becomes possible that configurations of which the trust rating has dropped get reconfigured automatically by their package managers by replacing one of the dependent components with a better trust rating, creating a safer SECO.

The package trust service will be somewhat intelligent: with several reproducible builds, low fix times, and few occurrences in Common Vulnerability and Exposure databases (CVEs), packages will have higher scores. This model will be designed such that it is extensible with new trust data if it becomes available. The scores will possibly be normalized to easily usable trust levels from Untrusted (T0) to Trusted (T5). One of the unique aspects of the TrustSECO is that it also uses the other package data to distribute packages over the trust levels evenly, to encourage package publishers to improve trust metrics continuously. Without action, their trust level can drop because the ecosystem is improving as a whole.

In Section 3, we will explore what data we can ethically gather on package software engineers, what data is available about project health.

Software Ecosystem Alert System
We build a tool that asks for data from the Trust Score Calculation Component and updates package providers and end-users on the status of their products and configurations. Additionally, the TrustSECO platform can be seen as an intrusion detection system, and if malware code is identified in a software package, we can warn both the end-users and the providers. TrustSECO uses all the data from Table 1 to secure the SECO, this can be done by running the Trust Reporting Service as a cron job on a user's system.

There are other services available that currently provide alerts to software end-users about vulnerabilities. For instance, many package managers provide some checkboxes on whether a package is vulnerable. One of the most notable commercial services is `Snyk.io`, which has developed its proprietary tools and database for vulnerability notification. While Snyk has been exceedingly accessible for such notifications, they insufficiently address the open-source community, its concerns, and its need for openness.

3 Trust Score Calculation

We provide information systems end-users with insight into the trust factors efficiently, by calculating a trust score for each information system and software package. Furthermore, package managers can use the trust scores to deploy reliable information systems automatically, so the trust scores should be readable by both humans and systems. The trust scores must have several properties. First, they should be *multi-dimensional*, i.e., they should give insight into the package, the package version, and the community of software engineers that produced it. Secondly, they must be *fully transparent*, such that any other stakeholder can calculate the same trust scores with the same data, which also helps build consensus about trust in the broader software engineering community. Thirdly, the scores must be *combinatorial*, in the sense that scores can be combined to calculate trust ratings for package compositions. Finally, the trust scores should be *numeric* to rapidly identify the trustworthiness of a software package version.

What are we scoring?
When we consider what can be scored to determine the software trust, we first confirm the trust factors through a series of research to determine the dimensions of the trust score based on these factors, however, we found most literature only cover SDLC or some perspectives throughout the package selection process, instead of providing a comprehensive metrics for the whole picture, and some metrics lack practicality cannot be collected easily. We consider trustworthiness attributes will be attributed to the following aspects:
Technical Factors. Quality attributes matter most when software end-users are accepting software [11]. Here, we consider the technical factors more about the quality attributes to measure the packages and versions from the perspectives of both functional and non-functional requirements, such as if the functions

align with users' requirements and original design, or if there are any outdated dependencies or numbers of open issues.

Also, we focus on vulnerability, as it negatively impacts software quality. If a vulnerability is detected in a package version, the trust referred to above is diminished appropriately, depending on the severity of the vulnerability. We are currently experimenting with a multiplier between 0 and 1 that diminishes the otherwise good trust score when a particular version contains a threatening vulnerability. In this way, other versions can keep a relatively high score, while this version is lowly rated. One of the significant upsides of TrustSECO taking care of this information is that the software end-users do not have to worry about keeping their systems up to date. If connected to a package manager that monitors the system, it can automatically reconfigure the system to include a version with a high trust score.

Ecosystem Factors. Ecosystem factors are not only the intrinsic properties to all stakeholders, but also reflect the relationship between end-users with software engineers, organizations, and communities [9]. Thus, first of all, we focus on the software engineers' experiences, knowledge, and skills; organizations or communities' reputation, popularity, and positive support; and interest from various stakeholders, e.g., packages' number of tags in Stack Overflow, or the number of downloads in npmjs. In addition, the cost is an important part of ecosystem factors, as one of our major goals in adopting software packages is to reuse the fulfilled program to speed up the coding process, and save costs. Here we emphasize the time and economic costs as the key elements to implement the software package, e.g. project time, budget, and software licenses.

A score with multiple dimensions

Based on various definitions of software trust and the SECO's context, we believe that software trust does not only talk about the software package itself but also a complex and comprehensive concept about all relevant impact factors. Score calculation can be unpacked through the following dimensions:

Packages + Versions. Building trust is an ongoing process that changes over time. The same package may have different trust scores in different versions. Packages' and package versions' trust scores are influenced by several factors, such as: (a) bug-fixing times, (b) end-users' experiences with specific package versions, (c) known vulnerabilities, (d) unreliable dependencies, (e) trust in the individual software developers.

Package Managers. Package managers managing a collection of software packages, are a part of the infrastructure that enables anyone to use the software and a SECO's backbone. Unfortunately, these package managers are not as secure as users think they are. At different stages in the software life cycle, vulnerabilities [1] can enter the software, and the package managers assume no responsibility for this. Therefore the end-users' concern for package managers should be on security and continuous support. TrustSECO cares about the following facts about package managers: (a) Usage frequency. (b) Any malicious,

outdated, or broken dependencies. (c) Reputation and popularity. (d) Compliance with security standards. (e) Recent compromises of the package manager.
Software Engineers. The software engineers who provide our society its critical information systems impact the trustworthiness of the packages. Factors that play a part in determining trust from individual engineers are: (a) The years of activity on well-known platforms such as Github, (b) Star ratings on development platforms, (c) Any negative experiences. We have to be careful about individuals, as TrustSECO should not function as a surveillance instrument. We do not want software engineers to get into trouble for, e.g., a bug that they introduced in an important package because their intentions were probably benign. The TrustSECO infrastructure will be developed in a way where trust facts can be used to determine the trust in a software package, but cannot lead to the individual trustworthiness of a software engineer decreased.
Software Organizations. Organizational behaviors show the ability to prevent risks and provide effective support, the following factors are considered: (a) Organizational support, e.g., fixing bugs on time and activity on the mailing list. (b) Popularity, reputation, and user reviews, e.g., number of watchers, number of contributors, user satisfaction.

4 Distributed Ledger Technology for TrustSECO

We select DLT as it enables the provision of trust knowledge as a "commons", i.e., a resource that can be generated and consumed by the community. The current vision is that the DL must be private, in order to guarantee the authenticity and validity of the data, as well as to guarantee the interests of all participants in the network. Therefore, we need to ensure that it is impossible to flood the DL with anonymous parties' trust facts.

The DL is filled by software packages providers and end-users to store events that contribute to or detract from the trust score, for instance, the confirmation of a reproducible build[5] by an end-user (positive) or the observation of an occurrence of a package version in a CVE database (negative). The trust data will be crawled and confirmed by the TrustSECO nodes, i.e., trusted nodes in the network that collaboratively confirm trust facts, even to the extent that they check for a reproducible build.

Trust facts that are mined from software repositories, gathered from users and end-user organizations and collected from third-party trusted databases. One of the challenges in this project is how we deal with *oracles* to obtain those data from the outside world to accommodate in our DL [10], i.e., data sources that can be considered as reliable or more reliable than TrustSECO, such as the CVE database, national vulnerability databases, and software repositories. For now, we will ensure that these databases are observed by multiple actors in the TrustSECO platform, but in the future, we envision creating a hard connection to these trusted systems.

[5] `https://reproducible-builds.org/`

The design of the ledger is non-trivial. We address three challenges. First, it is challenging to design or select the consensus algorithms that best fit the ledger. In particular, we will adopt the Raft [5] algorithm, to ensure that even under stress, we are guaranteed consensus. We will need to find a balance between costs to the network (of confirming data, for instance) and a healthy consensus size that leads to reliable facts in the ledger. Secondly, we need to explore what the best way is of handling the trusted third parties, such as Libraries.io and Github, who provide excellent sources for data, that is typically un-compromised. As well as the DL network must be confidential, which helps to eliminate some commercial interests from undermining trust. Finally, we need to design mechanisms that make the ledger sustained by the community, without incurring significant costs to the participants in the ecosystem.

Based on these challenges, the main requirements for the ledger are: (a) The DL must be scalable and cheap, (b) The DL must be easy and fast to search through, (c) The DL must be tamper-proof, and (d) The DL's transaction format can evolve, so new transaction types can be stored later.

Before the DL design, we first need to explore the data that is available about packages. There is a significant amount of trust data available online. For instance, we explore how existing CVE databases are used, or we can use digital signatures to ensure that a particular set of software engineers from a particular location have worked on source code. While this does not guarantee the reliability of the code, this kind of knowledge can be trusted by the software end-users. The sources of our trust data are twofold:

Online Data Sources. We will make an exploration of the data that is available about package reputation. For instance, what data we can ethically gather on package software providers, what data is available about project health, and whether we can support customers of packages to give trust ratings to packages. Our initial experiments will be primarily with the data available on Libraries.io and our own generated data. A successful build reproduction by a package end-user in the ecosystem, for instance, is a data point that adds to the trust in a particular package. A list of potential data sources are: (a) Github: project health, software developer identities, etc. (b)Libraries.io: stars, SourceRank, etc. (c) Common Vulnerabilities and Exposures Databases: (c.1) `https://www.vulncode-db.com/`, (c.2) `https://nvd.nist.gov/vuln/search`, (c.3) `https://cve.mitre.org/cve/`, etc. (d) Repology.org: Contains data on which package repository contains the most up-to-date package. Contains repository health statistics. (e) Owasp: Meta-data for what constitutes a trustworthy package. (f) Trustix: Reproducible build data.

Software Stakeholders. TrustSECO participants will benefit from using TrustSECO data, accordingly, they must also provide data themselves. If a community provides data about their satisfactory usage of a particular package, it will contribute positively to the trust score. Furthermore, if end-users are not happy with a particular package (version), it may detract from the trust score.

The TrustSECO DAO for a Sustainable Infrastructure
We need a sustainable infrastructure that can independently function after the academic project ends. As trust is a phenomenon created in the SECO as a form of *digital commons*, it should also be managed by the ecosystem. The open-source community has a track record of self-cleansing through transparency and communal solution finding, so we hypothesize that the ledger can sustain itself through a contribution from the community. We envision a Distributed Autonomous Organization (DAO) [13], i.e., an organization governed by its participants through rules and regulations formalized in smart contracts. The organization will have to determine different things, such as who can enter the network and in what conditions, and how the network resources need to be maintained.

To launch the DAO successfully, we take the DAO reference model of Wang et al. [13] as a starting point, who address the major design decisions as a choice in Organization Form, Incentive Mechanism, and Governance Operation. We discuss the best way to enter the DAO and make design choices with our industry partners on this basis. We aim to include incentives for participants to participate in the infrastructure, such as access to exclusive data when a participant provides hardware for the project.

5 Evaluation of the Proposed Platform and Future Work

Launching a platform such as TrustSECO requires extensive evaluation before we can assume success. As we follow a design science approach [4], we must continually evaluate our approach, our intermediate products, and our potential stakeholders. The aim is to make our artifacts might reasonably and scientifically by progressively improving and adapting the platform to meet the needs of the experts and stakeholders we meet [12]. This is done by broadcasting our ideas to the different communities that are working on software engineering, software security, information systems, and distributed ledger communities. We are in discussion, for instance, with the Lisk Center in Utrecht, where multiple DLTs are under development. Meanwhile, there are multiple international commercial stakeholders who have committed to this project with both research funding and time. And accordingly our artifacts are evaluated with each increment of the platform [12]. We are currently surveying commercial stakeholders and are exploring how such parties would be willing to participate in the TrustSECO platform.

Secondly, our artifacts are evaluated using technical experiments [12]. We will evaluate the technical performance with real-world data. This approach helps us to identify and address some potential performance problems such as response time, number of concurrent users, Transactions Per Second or Queries Per Second. For example, one concern is how our product handles the overhead of adding query DL to the ecosystem whenever someone builds a software system, especially when the system has dozens or even hundreds of components.

We define the following as near-future work. First, we will create a survey to establish how our stakeholders would calculate trust scores based on the impact

factors identified in the SLR. Secondly, we continue developing the prototype towards a minimum viable product for early experiments with existing package managers. Thirdly, we intend to deliver a prototype for the score calculation mechanism by the end of this year.

6 Conclusion and Status of the Project

This paper introduces the main concepts behind TrustSECO, a distributed ledger-based infrastructure that aims to provide a trustful SECO. TrustSECO targets the whole ecosystem instead of focusing on one language or package ecosystem alone. With TrustSECO, software package providers can guarantee that the software that has left software engineers' desks worldwide is the same as the software installed on the end-users' systems and that the software is maintained in a 'healthy' manner. Furthermore, TrustSECO provides users with insight into the trust levels that they can put into the software they use daily. Finally, we hope that TrustSECO can become the plumbing under the trust system that guarantees the software we use to run a modern society. In the next paragraphs, we report our progress.

Structured Literature Review into Software Trust - We have a team of five researchers who have recently finished an SLR to determine how trust in software is defined and the factors that positively or negatively influence the perception of trust in the SECO by studying 593 scientific articles. We formulated the following research questions to guide our study: (1) How is the concept of software trust defined in literature? (2) What trust factors do end-user organizations consider when selecting software products? We uncover the trust relationships between end-user organizations and (a) software producing organizations, (b)software products and versions, (c)software developers, (d)software package managers.

TrustSECO Infrastructure Prototype - We have developed a prototype of the TrustSECO infrastructure[6]. The prototype works as a plug-in for the npm and tells npm to avoid particular packages or package versions based on the trust data that we have stored in the DL. The DL is a database of trust facts that is currently stored in the Ethereum blockchain, but we are looking for another platform for reasons of scalability [3].

Interview Study with Software Engineers - We conducted structured interviews with twelve software engineers from different domains to explore how they perceive trust during the software package selection [7]. From the interviews, we conclude three things. (1) Technical factors and ecosystem factors are equally important. (2) Documentation, e.g. testing reports, training material, or even grey literature, can give a first impression of the software's technical capabilities. (3) Trust is subjective. Different kinds of software engineers hold different views on trust. Hence, provision of specific guidelines, reliable evidence, and customized functionalities are needed to help end-users and organizations build trust scores.

[6] `https://github.com/SecureSECO/TrustSECO`

Business Model Exploration - We have received a research grant to explore the business model that can sustain the TrustSECO infrastructure after the academic project is finished. We are using the DAOCanvas[7] to explore how we make the TrustSECO infrastructure sustainable and self-governing with a DAO.

Acknowledgements - We wish to thank the TrustSECO team that participated in the Odyssey Momentum Hackathon for their conceptual contributions to this paper. Specifically, we want to thank Tom Peirs, Jozef Siu, Venja Beck, Floris Jansen, and Elena Baninemeh for their inspirational ideas and their code on `https://github.com/SecureSECO/TrustSECO`.

References

1. Cadariu, M., Bouwers, E., Visser, J., van Deursen, A.: Tracking known security vulnerabilities in proprietary software systems. In: 2015 IEEE 22nd Int. Conference on Software Analysis, Evolution, and Reengineering. pp. 516–519. IEEE (2015)
2. Cho, J.H., Chan, K., Adali, S.: A survey on trust modeling. ACM Computing Surveys (CSUR) **48**(2), 1–40 (2015)
3. Farshidi, S., Jansen, S., España, S., Verkleij, J.: Decision support for blockchain platform selection: Three industry case studies. IEEE TEM (2020)
4. Hevner, A.R., March, S.T., Park, J., Ram, S.: Design science in information systems research. MIS quarterly pp. 75–105 (2004)
5. Howard, H.: Arc: analysis of raft consensus. Tech. rep., University of Cambridge, Computer Laboratory (2014)
6. Iqbal, M., Matulevičius, R.: Blockchain-based application security risks: a systematic literature review. In: International Conference on Advanced Information Systems Engineering. pp. 176–188. Springer (2019)
7. Jansen, F., Jansen, S., Hou, F.: Trustseco: An interview survey into software trust. arXiv:2101.06138 (2021), `https://arxiv.org/pdf/2101.06138.pdf`
8. Jansen, S., Cusumano, M.A., Brinkkemper, S.: Software Ecosystems: Analyzing and Managing Business Networks in the Software Industry. Edward Elgar (2013)
9. Larios Vargas, E., Aniche, M., Treude, C., Bruntink, M., Gousios, G.: Selecting third-party libraries: The practitioners' perspective. In: Proc. of ESEC/FSE. p. 245–256 (2020)
10. Lo, S.K., Xu, X., Staples, M., Yao, L.: Reliability analysis for blockchain oracles. Computers & Electrical Engineering **83**, 106582 (2020)
11. Paulus, S., Mohammadi, N.G., Weyer, T.: Trustworthy software development. In: IFIP Int. Conf. on Comm. and Mult. Security. pp. 233–247. Springer (2013)
12. Peffers, K., Rothenberger, M., Tuunanen, T., Vaezi, R.: Design science research evaluation. In: International Conference on Design Science Research in Information Systems. pp. 398–410. Springer (2012)
13. Wang, S., Ding, W., Li, J., Yuan, Y., Ouyang, L., Wang, F.Y.: Decentralized autonomous organizations: concept, model, and applications. IEEE Transactions on Computational Social Systems **6**(5), 870–878 (2019)
14. Zhu, M.X., Luo, X.X., Chen, X.H., Wu, D.D.: A non-functional requirements trade-off model in trustworthy software. Information Sciences **191**, 61–75 (2012)

[7] `https://daocanvas.webflow.io/`