Security and Privacy Threat Analysis for Solid

1st Omid Mirzamohammadi
COSIC, KU Leuven
Leuven, Belgium
omid.mirzamohammadi@esat.kuleuven.be

2nd Kristof Jannes
imec-DistriNet, KU Leuven
Leuven, Belgium
kristof.jannes@kuleuven.be

3rd Laurens Sion
imec-DistriNet, KU Leuven
Leuven, Belgium
laurens.sion@kuleuven.be

4th Dimitri Van Landuyt
imec-DistriNet, KU Leuven
Leuven, Belgium
dimitri.vanlanduyt@kuleuven.be

5th Aysajan Abidin
COSIC, KU Leuven
Leuven, Belgium
Aysajan.Abidin@esat.kuleuven.be

6th Dave Singelée
COSIC, KU Leuven
Leuven, Belgium
dave.singelee@esat.kuleuven.be

***Abstract*—This paper provides an in-depth security and privacy analysis of the Solid protocol. Solid is a specification that allows user data to be stored decentralized in a personal online datastore (pod) independent from the application. This allows users to easily migrate to a different service and have more control over who data is shared with. We provide a comprehensive overview of the authentication, identification, and authorization protocols within Solid. We make use of the SPARTA threat modeling tool to assess the security and privacy aspects of Solid by modeling a realistic finance analytics application envisioned by the Solid community. This concrete use case allowed us to prioritize the residual threats in Solid. We employ methodologies such as STRIDE and LINDDUN for robust security and privacy threat modeling. The findings highlight the existence of several critical threats in the Solid specification. This is especially the case for privacy threats, which although it is an essential aspect of Solid, has so far not yet received enough attention, as our results indicate. These findings can be employed in future work to prioritize which residual threats to address and mitigate first.**

***Index Terms*—Solid, Security, Privacy, STRIDE, LINDDUN, SPARTA**

I. Introduction

The traditional centralized architecture of the web has led to a reality in which a few service providers are continuously harvesting the personal data of their users, while these users tend to keep using these service providers due to lock-in, convenience, and the network effect. However, awareness is growing about the possible individual and societal impacts of such centralization of online data-centric activities, and service providers, individual users and society at large are increasingly aware of possible negative outcomes.

According to Tim Berners-Lee, the inventor of the web, there are some key challenges in the traditional structure of the web [1]. Firstly, users have no control over their data. By entrusting their personal information to service providers, users relinquish the ability to conveniently manage, share, correct, delete, or migrate their data as desired. Furthermore, by having access to large volumes of personal data, service providers are becoming very effective in personalizing information through advanced analytics techniques that involve learning and predicting user preferences through similarity-based analysis, etc. Service providers can exploit this information to manipulate or influence the behavior of users. Armed with an understanding of users' preferences and interests, providers may disseminate misinformation for financial or political motives, eroding trust and undermining the integrity of information sources. Consequently, since there are a few renowned platforms, people might be targeted by some of this fake news for political advertisement purposes.

Moreover, the ever-growing user base poses a challenge for service providers as the volume of data they must collect becomes unmanageable within the confines of legitimate limitations [1]. Consequently, transitioning to a structure that eliminates the dependency on accumulating vast amounts of data becomes appealing for both users and potential service providers. Such a transformation would shift competition toward innovation rather than data acquisition.

Solid, introduced in 2016 [2], presents an alternative web architecture that addresses these concerns. In Solid, users' data is stored in Personal Online Data stores (pods) rather than being stored on the servers from the service providers. This paradigm allows users to exercise granular control over access permissions, enabling them to choose which entities within the system can access their data and revoke access whenever desired. Additionally, this decentralized approach makes switching between different applications more seamless for users. Notably, the advantages offered by Solid have captured the attention of certain governments, such as Flanders [3], who are actively pursuing its adoption.

A more fundamental form of decentralization however does not automatically alleviate security and privacy concerns which are diverse and broad. A fundamentally different architecture brings a number of unknown risks related to security and privacy [4]. While a number of security controls have been integrated into the Solid technology stack, an in-depth end-to-end analysis of the potential threats and risks is currently still generally lacking. This is however an important prerequisite for a gap analysis and systematic mitigation of the diverse concerns related to security and privacy.

In this paper, we present an in-depth security and privacy analysis of the Solid technology. The methodology involves extensive threat modeling and analysis (i.e. assessment of possible threat scenarios on a system model) of a representative and real-world Solid application case: analytics based on personal finance information across a larger set of data subjects. We apply both STRIDE [5] for security and LINDDUN [6] for privacy threats. The performed analysis is not conducted agnostic of existing Solid security and privacy solutions and involved a thorough modeling of the identification, authorization and authentication protocols in Solid. To ensure thoroughness, reproducibility and explainability of the outcomes, we adopt SPARTA[1] [7] which performs automated and generative threat elicitation and performs risk assessment based on a wide variety of factors such as an assessment of asset values, the strength and application of security solutions, the characterization of attacker capabilities, etc. Applying the analysis to a concrete and realistic Solid use case (the financial analytics case) allows us to express risk outcomes in a more concrete and tangible manner, and also enables us to validate the outcomes in the specific context of the application. The application case itself is representative of a larger class of use cases in which personal data is collected from many individual data subjects, it is aggregated and analyzed and then used in support of a commercial offering.

An in-depth overview of the authorization and authentication protocols employed in the Solid architecture has been provided in this paper. The contributions of this paper can be summarized as follows: (i) we perform an in-depth threat analysis focusing both on the security and privacy residual risks of Solid's authentication protocol using state-of-the-art threat modeling tools; and (ii) we leverage the outcomes analysis to identify critical threats for each data flow and entity within the Solid ecosystem.

The rest of this paper is structured as follows. Section II provides an overview of Solid and explains the finance use case. Then, in Section III, we explain the background of our threat modeling methodologies and the threat modeling tool we employed. Section IV explains how we modeled the finance application of Solid in SPARTA. Section V presents our results from SPARTA's analysis. Finally, Section VI summarizes our main results and discusses future work based on our paper's results.

II. Overview of Solid

This section provides the necessary background on the Solid platform and its security architecture. Section II-A first outlines the different entities involved in Solid. Then, Sections II-B to II-D discuss the security and privacy solutions that already make part of Solid, respectively for identification, authorization and authentication. After outline the generic architecture of Solid in Section II-E, the motivational case –a financial analytics application– for this article is discussed in Section II-F.

A. Entities

Basically, there are five types of entities in Solid:

Agents	Agents encompass the interfaces of users who desire to store their data and exert control over its access.
Web application	These entities use agents' data, subject to their consent, to deliver various services. Additionally, agents can delegate the authentication process to web applications, enabling access to other agents' data, provided that the necessary access permissions have been granted beforehand.
Pod provider	Pod providers are responsible for providing pods for the agents. Upon receiving valid queries from the agents or the web applications, they should send the required data.
Identity provider	These entities authenticate agents, allowing pod providers to validate the authenticity of agents' requests. They can also maintain access control lists for agents, specifying which agents are authorized to access their data. Notably, a pod provider can also function as an identity provider.
Certificate Authority	To make verification of the providers possible for the agents and the web applications, the certificate authority issues certificate for the providers.

In addition to the aforementioned entities, we can have entities called **aggregators** that are responsible for collecting data from various pods for analytical purposes and collecting the aggregated data in some pods. Like the agents and the web applications, they rely on their identity providers for authentication and authorization. We consider these kinds of entities in our threat modeling.

B. Identification

Solid employs the WebID protocol [8] for identifying the agents and the web applications. A WebID is a URI (Uniform Resource Identifier) denoting an agent. Each WebID of an agent is associated with one WebID profile document and is under the control of that agent. The WebID profile document consists of RDF (Resource Description Framework) graphs using Turtle language [9]. An RDF graph has three components; subject, predicate, and object. The predicate describes the relation between the subject and the object. For instance, if a WebID is associated with a public key PK, the subject, predicate, and object would be the WebID, public key, and PK, respectively. The WebID for each agent will be generated by the identity provider after an authentication process. This process could be an Email verification like the existing pod provider (in developer preview) Inrupt Pod Spaces.[2]

C. Authorization

Web Access Control (WAC) [10] is being used for authorization in Solid. Using an Access Control List (ACL) ontology, agents can define an access control policy over their data. These policies can be defined by RDF graphs specifying who can access a document and what their access mode is, i.e., what they can do with this document (read, write, append,

[1]https://sparta.distrinet-research.be/

[2]https://start.inrupt.com/profile

or control). It is exemplified (not imposed) in [10] that we can have authorization servers storing these RDF documents. Therefore, receiving an access request, they can search the RDF document by running a SPARQL query code [11]. Then depending on the value of the boolean output of the code, they respond with an access token or an access denied message.

D. Authentication

At first, the WebID-TLS protocol [12] was proposed for the authentication in Solid [13]. However, it is not employed anymore (it can be used as an additional authentication method) [14]. Instead, the Solid-OIDC protocol [15] is being used in Solid.

Below, the protocol is explained with a scenario in which Alice accesses a document from Bob (assuming Bob has already granted access to Alice). In this scenario, Solid-OIDC supports authentication delegation so that Alice can access Bob's document using a web application. The protocol is divided into two phases.

In the first phase (Fig. 1), Alice's interface will delegate the authentication process to a client (web application). This phase consists of five entities. Alice's interface is the agent, and the client is the web application. A client's ID and an end-user's (Alice'e) WebID document can be located in the servers of the client's and Alice's identity providers, respectively. We also have an OpenID provider, which is an identity provider of Alice and has issued Alice's WebID. This phase is very similar to the OpenID Connect flow called the Authorization Code Flow with PKCE. In what follows, we briefly explain how this phase works.

1) Alice's interface sends its OP's (OpenID Provider's) URL (Uniform Resource Locator) or WebID to the client.
2) If the WebID was sent, the client retrieves the OP's information from Alice's profile.
3) The client retrieves OP's configuration to get information about the authorization process.
4) Then, the client will send an authorization request to the OP along with a commitment. this commitment is the hash of a secret value.
5) The OP will retrieve the client's profile from its ID document to verify its URI.
6) Afterwards, Alice will be directed to the log-in web page of the OP.
7) Authenticating Alice's identity, the OP will generate a cryptographic random string as the authorization code.
8) Firstly, the client generates an asymmetric key pair. Then by employing a hash function, digital signature, and the authorization code, it sends a token request to the OP. This message contains the reveal of the fourth message (the input of the hash function). This ensures the OP, that the sender of the eighth and fourth messages are the same.
9) If the OP verifies everything in the eighth message, it generates a token for the client. This token states that the client with a specific ID and public key (generated before sending the eighth message) is authorized to represent Alice in requesting Bob's data.

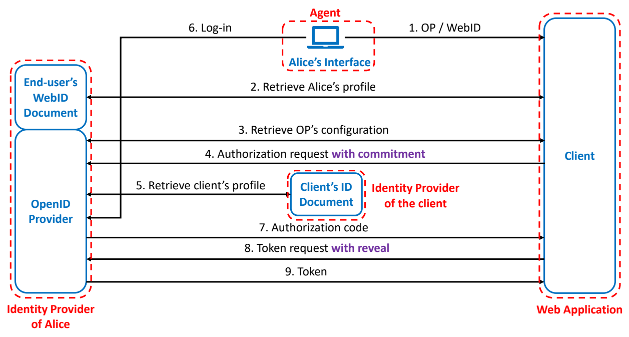
Agent
Alice's Interface
6. Log-in
1. OP / WebID
End-user's WebID Document
2. Retrieve Alice's profile
3. Retrieve OP's configuration
4. Authorization request with commitment
Client
OpenID Provider
5. Retrieve client's profile
Client's ID Document
Identity Provider of the client
7. Authorization code
8. Token request with reveal
9. Token
Identity Provider of Alice
Web Application

Fig. 1. The First Step of Solid-OIDC [15]

In the second phase (Fig. 2), the client will use the token it received from the OP of Alice to access Bob's document. In this phase, we have two new entities. The first one is a resource server which has Bob's document. This is the pod provider of Bob in our definition. The second one is Bob's authorization server which contains all the policies that Bob has set for his documents. Since these policies are over Bob's WebID, and each WebID can have multiple pods, it is logical to assume this authorization server is located in the servers of Bob's identity provider.

1) Firstly, the client will ask the resource server, which authorization server it should communicate with.
2) Finding out the authorization server, the client retrieves the configuration to get information about the authorization process.
3) Then, by using the token from the first phase, it requests an access token it can use for accessing Bob's document.
4) The authorization server extracts Alice's WebID from the token. Then it retrieves Alice's profile document to verify the validity and whether the claimed WebID's OP in the token matches the OP in the profile.
5) Then, the authorization server retrieves the OP's configuration, including its public key. By using all the information, it can verify the validity of the token.
6) If the token is valid and there is an RDF graph stating that Alice is authorized to have access, the authorization server will send an access token to the client.
7) The client requests Bob's document using the access token.
8) The resource server returns the result if the token is valid.

E. The Solid architecture

In Fig. 3, a simple structure of the Solid protocol is illustrated. The agents send their data and delegate the authentication process to the web applications and in return will be provided with some services. The agents and the web applications will follow some authentication and authorization processes to receive their WebID or their requested token

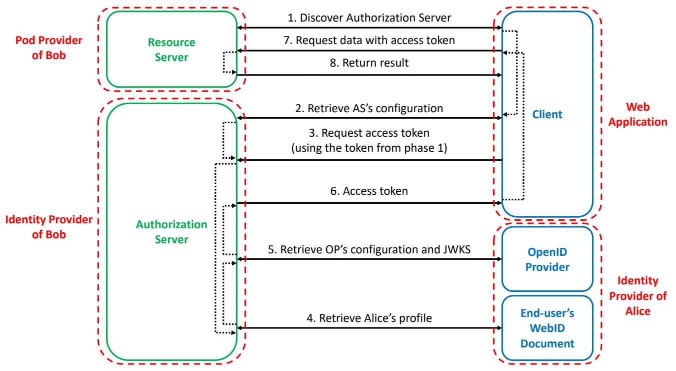
Pod Provider of Bob
Resource Server
1. Discover Authorization Server
7. Request data with access token
8. Return result
2. Retrieve AS's configuration
3. Request access token (using the token from phase 1)
6. Access token
Client
Web Application
Identity Provider of Bob
Authorization Server
5. Retrieve OP's configuration and JWKS
OpenID Provider
Identity Provider of Alice
4. Retrieve Alice's profile
End-user's WebID Document

Fig. 2. The Second Step of Solid-OIDC [15]

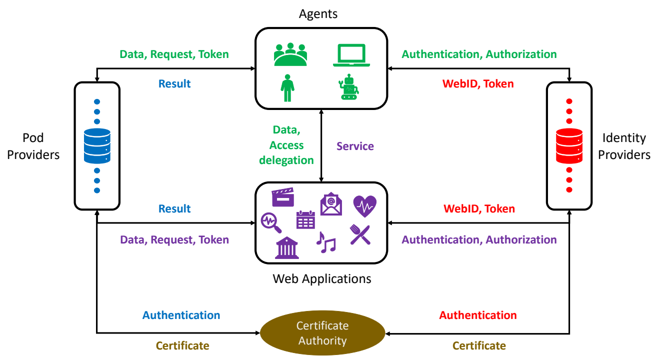

Fig. 3. Simple Architecture of Solid

from the identity providers. Then, the agents and the web applications can use the tokens to send their data or access requests to the pod providers and receive results. However, we did not mention the certificate authority in this structure. Using the mentioned protocols, the identity providers and the pod providers can authenticate the agents and the web applications. But it seems that the agents and the web applications cannot authenticate the providers. However, they can do that because the authentication protocols that we discussed are based on the HTTPS (Hypertext Transfer Protocol Secure) protocol which uses the TLS (Transport Layer Security) handshake. In the TLS handshake, a server should send its valid certificate (issued by a certificate authority) to the user [16]. Therefore, the agents and the web applications can verify the certificate of the providers before authenticating themselves.

F. Motivating example: Financial analytics case

To be able to correctly model all security and privacy threats of the Solid ecosystem, we will use a realistic use case as a concrete example. The Solid Financial Analytics case is a personal finance application for end-users. In its simplest form, this comprises of a user, a solid pod and the application itself. Users can enter all of their financial transactions into the application, which will store all this data in the Solid pod of the user. With this data, the application can show relevant statistics and insights into the users' financial situation. Users on the other hand know that their financial data is safe, as this data is never stored with the application itself. They have full control over their data pod and can withdraw access to this data whenever they want. These kind of applications on top of Solid already exist today.[3] However, by using the Solid concept of an aggregator, this use case can be enhanced. The aggregator will collect data from all Solid pods that it get authorization for to calculate aggregate statistics over the financial data from the users. This aggregate data can in turn be stored in a Solid pod from the aggregator so that other applications can access this data and display it to their users. Instead of users manually entering their financial transactions into an application into their Solid pod, financial institutions (e.g. banks) can provide this data for the user into their own Solid pod. This makes it easier for users, as their data in the pod is populated automatically, but also ensures that the data in the pod is correct, for example by requiring the financial institutions to sign this data. This way, aggregators can know they are working on correct data.

This example use-case is ideal for our threat modeling, as this case contains many different features and components of Solid: pod providers, identity providers, aggregators, 3rd party data sources (the financial institutions). It also handles highly sensitive financial data with potentially great impact for the user when threats are present. No user wants to expose their current financial situation or spending habits to any of the parties inside or outside the system.

III. Background on Threat Modeling

In this section, we first explain STRIDE and LINDDUN, which are the security and privacy threat modeling methodologies we used for threat modeling. Then we introduce SPARTA which is an automated risk-based threat analysis tool.

A. Security Threat Modeling Methodology

We use STRIDE for security threat modeling methodology. STRIDE threat modeling, developed by Microsoft [5], [17], offers a comprehensive and structured approach to identifying and addressing potential security threats by analyzing how an adversary may exploit vulnerabilities to software systems. STRIDE is an acronym of the six different threat types that a system can be endangered by:

S1. **S**poofing: Posing as somebody or something else;
S2. **T**ampering: Modifying the data without authorization;
S3. **R**epudiation: The possibility of denying performing an action that should be linkable to its performer;
S4. **I**nformation Disclosure: The disclosure of the information to unauthorized individuals;
S5. **D**enial of Service (DoS): Making the system inaccessible to its valid users;
S6. **E**levation of Privilege: Gaining more rights than were granted before, without authorization.

[3]https://budgetimize.com

B. Privacy Threat Modeling Methodology

LINDDUN [6] is used for privacy threat analysis. LINDDUN is a comprehensive and structured approach to identify potential privacy threats to personal data. LINDDUN is an acronym representing seven different types of threats[4]:

P1. **L**inking: Linking threat means connecting related data items to gain deeper insights into individuals or groups.
P2. **I**dentifying: Undesired learning the data subject's identity through leakage, deduction, or inference means identifying threat.
P3. **N**on-Repudiation: Non-repudiation threats arise when an individual is unable to refute specific claims or actions.
P4. **D**etecting: A system is vulnerable to detecting threats if unauthorized entities can identify data subject involvement, membership, or participation to the system.
P5. **D**ata Disclosure: Data disclosure threats refer to situations where personal data is exposed due to unnecessary data collection, processing, or involvement of unwanted parties.
P6. **U**nawareness and Unintervenability: This threat refers to inadequately informing, including, or empowering a data subject in its role and relation to the system.
P7. **N**on-Compliance: Not following the legislation (such as the GDPR [18]), inadequate personal data management, and insufficient risk management can lead to this threat.

C. SPARTA

The security and privacy threat analysis of the application is performed using the SPARTA threat modeling tool [7]. SPARTA analyzes a Data Flow Diagram (DFD) model of the application to automatically elicit security and privacy threats according to STRIDE and LINDDUN by iteratively going over every interaction in the model. Additionally, SPARTA performs a per-threat risk analysis to prioritize the identified threats. In addition to the DFD model, SPARTA leverages two key additional inputs in order to prioritize the security and privacy threats it elicits.

First, the DFD model in SPARTA is extended with support for modeling security and privacy solutions [19] to capture more comprehensive combinations of security and privacy countermeasures that affect multiple DFD elements. These solutions are taken into account in the subsequent risk analysis of the threats.

Second, the DFD model in SPARTA is enriched with asset values and data types. These values can be used to express which elements handle important information or sensitive personal data. This additional information is taken into account as part of the risk analysis (together with the aforementioned countermeasures) in order to calculate the residual risk value for every elicited threat. This way, threats for which there are already sufficient mitigations in place will have a reduced risk value and will receive a lower priority.

It is a recurring challenge in risk analysis to establish meaningful numerical values, especially when dealing diverse risks. SPARTA's risk model is based on FAIR [20] and supports expressing uncertainty in the provided values expressed as parameters for a modified-PERT distribution [21] (minimum, mode, maximum, and a confidence). SPARTA's risk calculation mechanism employs sampling of diverse distributions and takes into account many risk components, such as an expression of asset values, a characterisation of solutions, attacker profiles, etc. We refer the interested reader for more details about the risk calculation mechanism to [22].

The values reported by SPARTA are not absolute but relative values, in the sense that they can be used to directly compare individual risks and threats (also between projects), but also for example to monitor risk reduction over time.

IV. Modeling the Finance Application of Solid Using SPARTA

In this section, we first discuss our trust model. Then, we explain how we performed an in-depth security and privacy threat analysis of Solid, through its individual steps: (i) designing the DFD model based on the finance analytics use case, (ii) enriching and including the different security and privacy solutions that are already present in Solid, and (iii) assigning asset and risk estimates to the elements of the DFD model.

A. Trust Model

In order to have a sound and reproducible analysis of the Solid, it is also important to establish and document our trust model (assumptions and contextual considerations relevant to the analysis) up front. We adopt the core assumption that the entities involved in the system (the identity and pod providers) are semi-honest, i.e., they do their job well (they will not act maliciously), but they may be curious about the data they collect, for example about the individual data subjects (identity, role, involvement), about competitors in the ecosystem, etc.

B. The DFD Model

We designed the DFD model of the authentication protocol in a finance application of Solid. We assumed that there are two users (users 1 and 2) in the system with issued WebIDs from different identity providers (identity providers 1 and 2 respectively). They have already sent their ACL commands to their identity provider. They want to have access to their finance information using two different applications. These finance information has been sent to the users' pod providers (pod provider 1 and 2 respectively) by a financial institution. There is also an aggregator with its identity provider (identity provider 3) and its pod provider (pod provider 3). It was assumed that each of this users has already granted access to this aggregator to access their data to be able to access the statistical data it sends to pod provider 3. This access policy has been set in identity provider 3. Each provider has a controller and a data store. We provided the DFD model for a case that user 1 wants to see its own finance data, user 2 wants to see the statistical data, and the aggregator wants to receive the users' data.

[4]https://linddun.org/

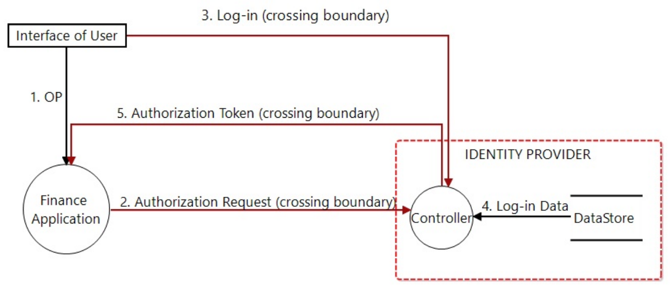
3. Log-in (crossing boundary)
Interface of User
1. OP
5. Authorization Token (crossing boundary)
IDENTITY PROVIDER
Finance Application
2. Authorization Request (crossing boundary)
Controller
4. Log-in Data
DataStore

Fig. 4. Authentication Delegation

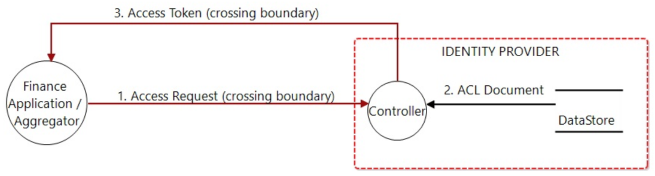
3. Access Token (crossing boundary)
IDENTITY PROVIDER
Finance Application / Aggregator
1. Access Request (crossing boundary)
Controller
2. ACL Document
DataStore

Fig. 5. Access Control

Since the whole DFD model is too big to provide in this paper, we will focus on the four main procedures separately:

1) Authentication Delegation: In this procedure the users delegate the authentication to a finance application illustrated in Fig. 4. In this figure, we basically simplified the first procedure of Solid-OIDC by omitting retrieving profile messages and merging the fourth, seventh, eighth, and ninth messages in Solid-OIDC into two messages between the application and the identity provider. There is one additional data flow between the data store and the controller of the identity provider, which is used for the log-in verification. In the case where the finance application wants to access its user's pod, the authorization token can also be used as the access token.

2) Access Control: This procedure is a simplification of the second step of Solid-OIDC as shown in Fig. 5. In this procedure, either the finance application or the aggregator sends an access request to the identity provider. Then, the identity provider looks into its data store to check whether the aggregator or the user who delegates the authentication process to the finance application has been granted to access the requested data before. If it has been granted, the identity provider issues an access token and sends it to the requester.

3) Access Request: Figure 6 illustrates the access request procedure. The finance application or the aggregator who has received the access token can use this to send an access request

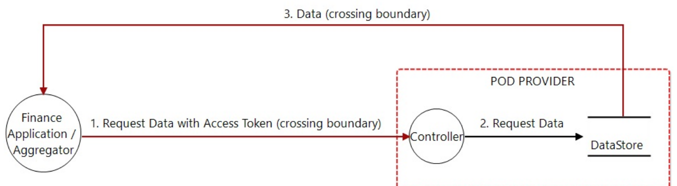
3. Data (crossing boundary)
POD PROVIDER
Finance Application / Aggregator
1. Request Data with Access Token (crossing boundary)
Controller
2. Request Data
DataStore

Fig. 6. Access Request

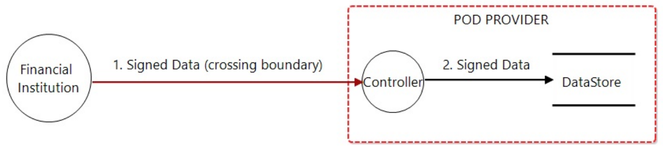

Fig. 7. Sending Data

to the pod provider where the requested data is located. The pod provider checks the validity of the access token and then will send the data to the requester.

4) Sending Data: In this procedure, the financial institution sends the signed finance data of the users to their pods. Before storing the data, the pod provider checks the validity of the signature. This procedure can be seen in Fig. 7. By employing this procedure, tampering threats located on the receiver of the signed data are mitigated.

C. Security and Privacy Solutions

We added two types of solutions to the SPARTA model. The first type includes the solutions that are inherent in the DFD model. However, the second type is associated with Solid's protocols. For modeling these solutions, we assumed that Solid protocols (for the identification, the authorization, and the authentication) are able to meet their goals. In what follows we discuss both of these types.

1) Within Trust Boundary Solutions: All the providers have two entities within their trust boundaries, the controllers and the data stores, communicating with each other. Therefore, all the data flows within these boundaries do not have any security or privacy threats.

2) Solid Solutions: There are five types of solutions added to the DFD model that are related to Solid:

- Secure Pipe: This is one of the predefined solutions in SPARTA. Countermeasures of this solution are like what HTTPS protocol offers. We added this solution to the "Log-in" data flows. Through encryption, it effectively mitigates the 'Information Disclosure' threat. Additionally, the presence of a certificate in the TLS handshake enables the sender to verify the identity of the receiver, thereby eliminating the "Spoofing Identity" threat on the receiver's side.
- Secure Pipe with Client Authentication: We added this solution to all the remaining data flows because, in all of them, the receiver can also check the identity of the sender. Therefore, in addition to the "Secure Pipe" countermeasures, there is no "Spoofing Identity" threat on the sender side. Moreover, as the receivers are able to establish a connection between a received data flow and its sender, the "Repudiation" threat on the sender side is also mitigated.
- Authentication Delegation: This solution is based on what the first procedure of the DFD model provides. Using this procedure, the user and the finance application can be authenticated by the identity provider. As a result,

TABLE I
OVERVIEW OF THE ASSIGNED RISK ESTIMATES AS INPUT FOR SPARTA.
EMPTY CELLS HAVE NO CORRESPONDING THREAT IN THE SPARTA OR LINDUN CATALOG.

DFD Element	S	T	R	I	D	E	L	I	N	D	D	U	N
User	H					H	L	L	H	L	L	H	L
Identity Provider	H	M	H	M	M	H	H	H	L	H	M	L	H
Pod provider	H	M	H	H	H	H	H	H	L	H	M	L	H
Application	H	H	L	H	H	H	H	H	L	L	H	L	H
Aggregator	H	H	M	H	M	H	H	H	L	L	H	L	H
Financial institution	H					H	L	L	L	L	L	L	H
Login				H	M		M	H	L	L	H		
IDP Info				M	M		M	H	L	L	H		
Authorization Request				M	M		M	H	L	L	H		
Authorization Token				M	M		M	H	L	L	H		
Access Request				M	M		M	H	L	L	H		
Access Token				M	M		M	H	L	L	H		
Request Data				H	H		M	H	M	M	H		
Data				H	H		H	H	H	H	H		
Aggregated Data				H	M		H	H	H	H	H		

the threats of “Spoofing Identity” and “Repudiation” are eliminated on their respective sides. Additionally, based on our assumption that the providers are semi-honest, neither the users nor the finance application can acquire additional privileges, thus mitigating the “Elevation of Privilege” threat. Moreover, the authorization token generated at the end of this procedure can not be tampered with. Therefore, we have “Tampering” mitigation on the finance application side.

- Access Control: This solution relies on the second procedure. Based on the reasons we discussed for the previous solution, by using this solution on this procedure, the “Spoofing Identity,” the “Repudiation,” and the “Elevation of Privilege” threats located on the finance application or the aggregator are mitigated.
- Access Request: This solution is added to the third procedure. Similar to the two previous solutions, effectively mitigating the threats of “Spoofing Identity,” “Repudiation,” and “Elevation of Privilege” present within the finance application or the aggregator. The access request includes a token generated by an identity provider, thereby mitigating “Tampering” on the controller side. Furthermore, the access token itself remains invulnerable to tampering, ensuring the absence of “Tampering” threats on the finance application or aggregator side.
- Signature: We added this solution to all the data flows containing the financial data or the aggregated financial data. By utilizing this solution, the “Tampering” threat on the receiver side is eliminated. Additionally, since the pod provider is semi-honest, and the financial institution is authenticated, there is no “Elevation of Privilege” threat located on the financial institution side.

D. Asset and risk estimates

To be able to prioritize any of the threats, SPARTA needs information about the potential loss that a threat can cause if it would occur. We provided this information both for the specific data assets, as well as an estimate for each different entity and data flow per STRIDE and LINDDUN threat. We choose to discretize these values in three categories: low (L) ($minimum = 0$, $mode = 1$ and $maximum = 2$), medium (M) ($minimum = 1$, $mode = 2$ and $maximum = 3$) and high (H) ($minimum = 2$, $mode = 3$ and $maximum = 4$). In the finance use-case, there are four different assets which are of value:

1) Financial data: this is the actual financial data of an individual user. This data is very valuable and its value is therefore estimated as *high*.
2) Aggregate financial data: these are aggregate statistics of financial data of many users, this can still be sensitive if data is not correctly aggregated with respect to privacy, i.e., too few users in one category. Therefore we estimate its value as *medium*.
3) WebID (and other URIs): this is an identifier of the user, we estimate it as less sensitive (*low*) compared to the actual financial data of a user, but it is still personal data.
4) Username and password: used by to user to authenticate with the identity provider. It is estimated as *high*.

For the different entities and dataflows in the DFD we estimate the risks as shown in Table I. These values are highly influenced by the type of data that is stored or processed at a certain entity.

V. RESULTS

In this section, we utilize the SPARTA analysis on our DFD model to assess threats and categorize them based on their risk value. Firstly, we evaluate the effectiveness of Solid’s solution in mitigating privacy and security threats. Next, we identify the highest-risk threats associated with each DFD element. Finally, we delve into the high-risk threats, providing a detailed explanation of their sources.

A. Impact of Solid’s Solutions in Mitigating Threats

Based on the assigned asset and risk values, SPARTA calculates the risks of each threat located in each DFD

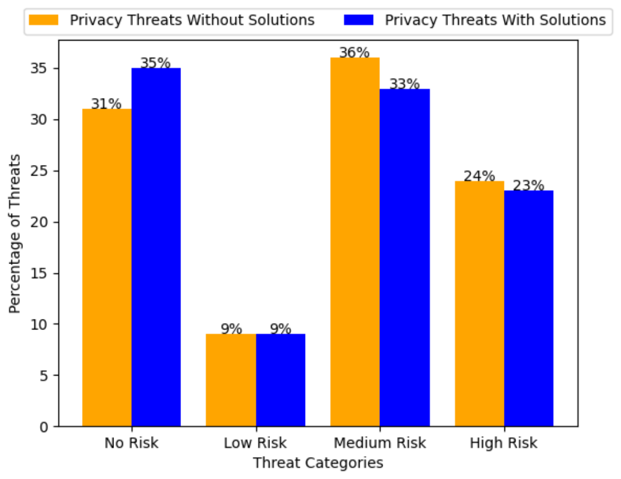

Fig. 8. Impact of Solid's Solutions on Privacy Threats

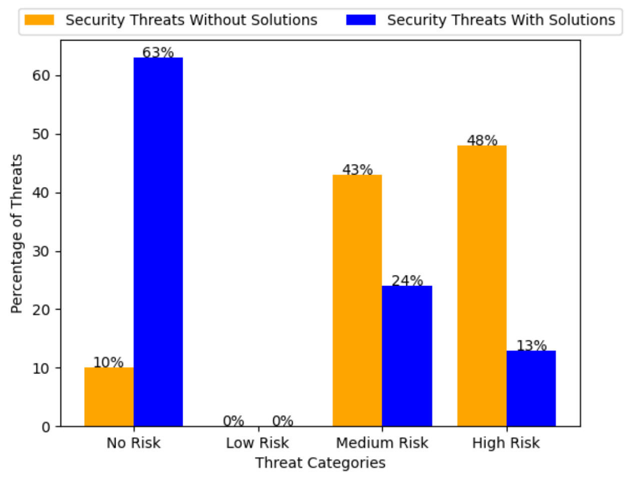

Fig. 9. Impact of Solid's Solutions on Security Threats

element. Upon analysis and prioritizing the threats based on their risk values, it was observed that the risk values could be categorized based on their proximity. It means that there are several noticeable gaps between these values that can be used for classification. Although the resulting risk values fluctuate whenever we re-run the SPARTA analysis as they are derived from statistical distributions, they always remain in the same category. Since the risk values fall within the interval of 0 to 16.4, we have divided this range into four equal parts using the risk values 4.1, 8.2, 12.3, and 16.4 as threshold margins. This classification results in four categories, namely "No Risk," "Low Risk," "Medium Risk," and "High Risk." For instance, all the threats with risk values between 4.1 and 8.2 are included in the "Low Risk" category. As all the threats' risk values in the first interval are zero, we call their category "No Risk."

To assess the impact of Solid's solutions on mitigating the threats we analyzed the DFD model again by removing these solutions from SPARTA. Therefore, only 'Within Trust Boundary' solutions were considered. In Fig. 8 and Fig. 9, the impact of Solid's solutions on the privacy threats and the security threats are illustrated, respectively. Solid's solutions effectively mitigate 54% of the security threats, indicating their significant impact on this category. However, when it comes to privacy threats, Solid's solutions only mitigate 4% of them. These figures highlight the varying impact of Solid's solutions on each threat category. Another noteworthy result is that by implementing Solid's solutions, the overall summation of risk values decreases by approximately 18%.

B. *Highest Risk Threats per DFD Element*

Table II illustrates the highest risk category located on each DFD element ("0", "L", "M" and "H" represent "No Risk", "Low Risk", "Medium Risk" and "High Risk", respectively). By examining the table, we can easily identify the level of risk posed by various threats on each element. For instance, when examining the identity providers, we observe that at least one of the non-repudiation and unawareness threats falls into the "Low Risk" category, while none of them are classified as "Medium Risk" or "High Risk." This table serves as a valuable tool for identifying the DFD elements that require more attention and prioritization for effective threat mitigation strategies.

C. *Unawareness and Non-compliance Threats*

Solid as a technology enabler is intended to improve data sovereignty, and puts a number of data management responsibilities into the hands of the data subjects. The predominant approach to attain this is the decentralized architecture of Solid and the degree and extent of controls offered to the data subject. Against this reality, it may seem counter-intuitive that Table II pinpoints a number of residual High-Risk Unawareness and Non-compliance threats.

The evaluation of relevant threats in the two final LINDDUN threat categories –Unawareness and Non-Compliance– in both cases relies heavily upon contextual information that is not represented in a data flow diagram (DFD) [23].

A DFD expresses data flow but not control flow, and thus the extent or nature of different controls offered to the data subject is more difficult to evaluate just at this basis. Furthermore, data subject controls were also not modeled as solutions (Section IV-C) to express the effect data subject control mechanisms would have on those data flows. Likewise, Non-compliance considers external legal aspects (e.g., '*is there an appropriate lawful ground for collection and processing?*'), and complementary perspectives on the system such as that of data management, e.g. '*how long is the data kept in the system? Is there automated removal or archiving, etc?*' and elements of the security architecture that may have privacy implications.

As neither types of information are represented in the DFD that was used for this analysis, the automated approach of SPARTA is currently not capable of correctly taking them into account. This outcome highlights and confirms the necessity to further improve the expressiveness of the DFD notation [24]

TABLE II
RESIDUAL RISK FOR THREATS LOCATED ON THE DFD ELEMENTS ACCORDING TO SPARTA.
EMPTY CELLS HAVE NO CORRESPONDING THREAT IN THE SPARTA OR LINDDUN CATALOG.

DFD Element	S	T	R	I	D	E	L	I	N	D	D	U	N
User	0					0	M	M	H	M	M	H	
Identity Provider	0	M	M	0	M	M	M	M	L	M	M	L	M
Pod provider	0	0	0	H	H	H	H	H	M	H	H	M	H
Application	0	0	0		H	0	H	H	M	M	H	M	H
Aggregator	0	0	0		H	0	H	H	M	M	H	M	H
Financial institution	0					0	M	M	M	M	M	M	
Login				0	M		M	M	M	M	0		
IDP Info							L	L	L	L	0		
Authorization Request				0	M		M	M	L	L	0		
Authorization Token				0	M		M	M	L	L	0		
Access Request				0	M		M	M	L	L	0		
Access Token				0	M		M	M	L	L	0		
Request Data				0	M		M	M	M	M	0		
Data				0	H		H	H	H	H	0		
Aggregated Data				0	M		M	M	M	M	0		

in function of automated threat analysis as performed by tools such as SPARTA.

D. *In-Depth Analysis of High-Risk Threats*

Having identified the DFD elements associated with threats in the "High Risk" category, it is important to determine the corresponding data flows. This understanding is crucial to see which part of the Solid protocol needs improvement to mitigate these threats. We do not discuss "Unawareness" and "Non-compliance" threats with "High Risk" in this part. In what follows we explain these threats for each DFD element in detail.

1) User: "Non-repudiation" threats apply to both data flows that users send, namely "Log-in" and "IDP info," as they can be traced back to a specific sender. Consequently, the users are unable to deny having sent these data flows at a later time.

2) Pod Provider: "Information Disclosure" is associated with the financial data stored in all data stores, as they lack encryption. "Denial of Service" threats are relevant to all data flows connected to pod providers, potentially causing users to be unable to access their financial data due to the single point of failure issue. The "Elevation of privilege" threat raises two concerns. Firstly, if pod providers are compromised, adversaries can elevate the privilege of any entity at will. Secondly, although pod providers should only collect users' data to send them when needed, having the ability to read these data leads to the "Elevation of Privilege" on their side.

Due to the collection of all users' data in plain text within the data stores of pod providers, and the association of data requests with unique WebIDs, we have "Linking," "Identifying," "Detecting," and "Data Disclosure" threats at the pod providers sides.

All the high-risk threats mentioned are associated with the data flows connecting to the pod providers, encompassing both the requests for data using access tokens and the financial data.

3) Application: The application holds a critical responsibility of receiving and managing crucial components such as the access token, authorization token, and financial data. Consequently, if the application is compromised by a "Denial of Service" threat, the users cannot access their data.

Given that applications have the ability to establish connections between user identities, their WebIDs, and associated data, there are high-risk threats of "Linking," "Identifying," and "Data Disclosure" threats at the application's side. These high-risk threats are associated with all data flows connected to this particular element.

4) Aggregator: "Denial of Service" threats are specifically linked to the access token and the data the aggregator receives. Hence, it is unrelated to the aggregated data.

"Linking," "Identifying," and "Data Disclosure" threats are located on the aggregators as they receive the plain-text data associated with some unique WebIDs. Moreover, all the requests are linkable to some unique WebIDs.

5) Data: As this element represents a data flow, it is susceptible to high-risk threats such as "Denial of Service," "Linking," "Identifying," and "Data Disclosure" wherever it is present.

VI. CONCLUSION

In this paper, we investigated the security and privacy threats in Solid using STRIDE and LINDDUN threat modeling methodologies in the SPARTA tool. We performed this analysis on Solid's Financial Analytics application case, allowing us to assign risk estimates to each element and asset to get the accurate importance of residual threats. The primary objective was to determine how much Solid already mitigates the threats and which remaining ones deserve more attention.

Our results show that the current Solid protocol specification mitigates many possible security threats. About half of all security threats are fully mitigated or irrelevant due to specific defenses inside the Solid protocol. However, several high-risk security threats remain, which Solid does not yet prevent in our trust model of honest-but-curious providers.

The results for privacy threats are different. Although privacy is an important motivator and selling point of Solid, little mitigations exist in today's Solid protocol. Only a few privacy threats are fully mitigated in Solid, and almost all LINDDUN threat categories apply to all medium or high-risk entities.

The findings presented in this paper identify the prioritization of critical threats that require further mitigation. We have also identified the locations of high-risk threats and explored the reasons behind these risks. The primary causes for all threats, except "Denial of Service," can be summarized as follows:

- The financial data being sent to the pod providers in a way that can be decrypted, which allows them to read the data.
- All communications and data collections within the Solid system being associated with unique WebIDs, which are linked to unique identities stored in the identity providers.

Therefore, given our assumption that the providers are semi-honest, they can undesirably gain information from the agents. The second cause highlights the need for privacy-enhancing technologies not only in the collected data but also in the identification and authorization protocols. Only by improving these protocols can we enhance the privacy aspect of the authentication protocol.

As a general guideline for Solid's developers, we propose the following ways to mitigate the critical threats in Solid.

- *Confidentiality*: The agent's (owner's) data should be encrypted to permit decryption solely by authorized agents and applications designated by the data owner. Attribute-based encryption (ABE) can be a viable option as it prevents pod providers from accessing their users' data while allowing the users to have control over who can decrypt their data.
- Privacy: All transactions and data within Solid are linked to unique WebIDs throughout the system, and these WebIDs are associated with the registration data of users maintained by identity providers. Consequently, if all providers collude, then the privacy objectives of this system could be compromised. Techniques like Anonymous Credentials and Zero-Knowledge Proofs can address these challenges.
- Availability: All the information of an agent, such as WebIDs and access control lists, is stored by a single identity provider. Consequently, decision-making during authentication heavily relies on the continuous availability of these identity providers. Similarly, one pod provider exclusively holds an agent's data (e.g., financial data). As a result, a compromise due to a DoS attack on any of these providers could lead to data loss or authentication issues for the affected users. To mitigate these risks, one can implement Threshold Authentication to distribute decision-making during authentication and employ Secret Sharing to distribute agents' data across different pod providers while securing the original data from the pod providers' direct access.

Our research serves as a road map for future endeavors, enabling the design of more privacy-preserving or cryptographic protocols fitting Solid's structure. These protocols can effectively mitigate the threats by first addressing threats with higher risk levels. We have shared the findings with the Solid team and are now focusing on developing novel security measures for future work.

Acknowledgment

This work is supported in part by SolidLab Vlaanderen (Flemish Government, EWI) and by CyberSecurity Research Flanders (Flemish Government, EWI).

References

[1] T. Berners-Lee, "Three challenges for the web, according to its inventor," March 2017. [Online]. Available: https://webfoundation.org/2017/03/web-turns-28-letter/

[2] E. Mansour, A. V. Sambra, S. Hawke, M. Zereba, S. Capadisli, A. Ghanem, A. Aboulnaga, and T. Berners-Lee, "A demonstration of the solid platform for social web applications," in *Proceedings of the 25th International Conference Companion on World Wide Web*, ser. WWW '16 Companion. Republic and Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, p. 223–226. [Online]. Available: https://doi.org/10.1145/2872518.2890529

[3] Digitaal Vlaanderen. The flemish data utility company. [Online]. Available: https://www.vlaanderen.be/digitaal-vlaanderen/athumi-het-vlaams-datanutsbedrijf/the-flemish-data-utility-company

[4] H. Halpin, "All that is solid melts into air: Towards decentralized cryptographic access control," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ser. ARES '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3538969.3544464

[5] M. Howard and S. Lipner, *The Security Development Lifecycle*. USA: Microsoft Press, 2006.

[6] K. Wuyts and W. Joosen, "Linddun privacy threat modeling: a tutorial." CW Reports, 2015.

[7] L. Sion, D. Van Landuyt, K. Yskout, and W. Joosen, "Sparta: Security & privacy architecture through risk-driven threat assessment," in *2018 IEEE International Conference on Software Architecture Companion (ICSA-C)*, 2018, pp. 89–92.

[8] A. Sambra, H. Story, and T. Berners-Lee, "Webid 1.0, web identity and discovery," March 2014. [Online]. Available: https://www.w3.org/2005/Incubator/webid/spec/identity/

[9] D. Beckett, T. Berners-Lee, E. Prud'hommeaux, and G. Carothers, "Rdf 1.1 turtle, terse rdf triple language," February 2014. [Online]. Available: https://www.w3.org/TR/turtle/

[10] S. Capadisli, "Web access control," July 2022. [Online]. Available: https://solidproject.org/TR/wac

[11] A. S. Eric Prud'hommeaux, "Sparql query language for rdf," January 2008. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/

[12] T. Inkster, H. Story, and B. Harbulot, "Webid-tls, webid authentication over tls," March 2014. [Online]. Available: https://www.w3.org/2005/Incubator/webid/spec/tls/

[13] A. V. Sambra, E. Mansour, S. Hawke, M. Zereba, N. Greco, A. Ghanem, D. Zagidulin, A. Aboulnaga, and T. Berners-Lee, "Solid: a platform for decentralized social applications based on linked data," *MIT CSAIL & Qatar Computing Research Institute, Tech. Rep.*, 2016.

[14] S. Capadisli, T. Berners-Lee, R. Verborgh, and K. Kjernsmo, "Solid protocol," December 2021. [Online]. Available: https://solidproject.org/TR/2021/protocol-20211217

[15] A. Coburn, elf Pavlik, D. Zagidulin, A. Migus, and R. White, "Solid-oidc," December 2022. [Online]. Available: https://solid.github.io/solid-oidc/

[16] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246 (Proposed Standard), Internet Engineering Task Force, Aug. 2008, updated by RFCs 5746, 5878, 6176. [Online]. Available: http://www.ietf.org/rfc/rfc5246.txt

[17] A. Shostack, *Threat modeling: Designing for security*. John Wiley & Sons, 2014.

[18] European Union, "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC," *Official Journal of the European Union*, vol. 59, no. L 119, pp. 1–88, May 2016.

[19] L. Sion, K. Yskout, D. Van Landuyt, and W. Joosen, "Solution-aware data flow diagrams for security threat modeling," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, 2018, pp. 1425–1432.

[20] J. Freund and J. Jones, *Measuring and managing information risk: a FAIR approach*. Butterworth-Heinemann, 2014.

[21] D. Vose, *Risk analysis: a quantitative guide*. John Wiley & Sons, 2008.

[22] L. Sion, K. Yskout, D. Van Landuyt, and W. Joosen, "Risk-based design security analysis," in *1st IEEE/ACM International Workshop on Security Awareness from Design to Deployment (SEAD)*, vol. 1. Institute of Electrical and Electronics Engineers, 1 2018, pp. 1–8. [Online]. Available: https://lirias.kuleuven.be/1656888

[23] D. Van Landuyt and W. Joosen, "A descriptive study of assumptions made in linddun privacy threat elicitation," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 1280–1287.

[24] L. Sion, K. Yskout, D. Van Landuyt, A. van Den Berghe, and W. Joosen, "Security threat modeling: are data flow diagrams enough?" in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 254–257.