Cover letter

We would like to thank the anonymous reviewers for their valuable comments that helped us to improve our paper. In addition, we would like to extend our gratitude to the conference chairs for giving us the opportunity to resubmit. In the following, we provide a summary of the main changes followed by actions taken based on each reviewer's comments.

- We have updated the Introduction section to highlight better our main contributions and the main reason/motivation for performing a threat modeling and postponing the risk assessment;
- Given the space constraint, we decided to remove the background section, as we introduce the Wallet ecosystem in detail in Section 2.1, and interested readers can refer to OID4VCI/OID4VP to deep dive into the specifications; Furthermore, we made some changes to the paper structure to make it easier for the readers to understand and gain enough space to add a discussion section to summarize our main results (Section 3);
- We have added the selection criteria for the threats, vulnerabilities, and controls in Section 2.2;
- We bring the complete list of protocol-specific threats (Table 2) and keep only the threats relevant to the context of EUDIW in the general threats table (Table 1);
- We added a discussion section (Section 3) to summarize the main results and lessons learned.

In the following, we detail how the individual comments have been addressed.

----------------------- **REVIEW 1** ---------------------

- The use of established frameworks like STRIDE and LINDDUN provides a thorough and structured approach to threat modeling, enhancing the credibility and depth of the analysis. However, the authors should explain why they selected these frameworks over others.
  - We have given the main reasons in Section 2.2, at the end of the threat category definition.
- While the paper identifies numerous controls, it offers limited discussion on the practical challenges of implementing these controls. Further evaluation of the feasibility and effectiveness of these controls in real-world scenarios would strengthen the paper. Classifying the vulnerabilities according to their feasibility and impact, as well as classifying the controls by their implementation difficulty and effectiveness in reducing the risk of the vulnerability, would be valuable.
  - We added a discussion in terms of implementation difficulty and effectiveness for a selection of six controls that have been selected based on the number of threats that they mitigate (see the new Section 3). Plus we added the activity related to a control-based risk assessment as future work.
- For future work, the article could benefit from a more detailed technical analysis of specific threats and how the proposed controls address these threats. This includes providing more concrete examples or case studies of potential attacks and defenses.

    - We added the activity related to a control-based risk assessment as future work.

----------------------- **REVIEW 2** ---------------------

- Besides the threat analyis presented authors fail to contribute to the research community with some practical results that could improve eIDAS in terms of security, privacy and GDPR compliance.
    - One of the contributions of this research is the identification of a set of security controls that aim to reduce the probability of threats and contribute to the security and privacy of the solution. We modified the contribution in the Introduction to emphasize our goals. Together with the list of threats and controls, we added a new section that discusses the main results and lessons learned (see Section 3).
- While, these aforementioned aspects are examined in the review section, it is not clear how they are considered in the merge of STRIDE and LINDDUN
    - Our approach of using STRIDE and LINDDUN had not the goal to merge them into a single unified framework but rather to map each identified threat to the most appropriate categories from both frameworks. This allowed us to address both security (via STRIDE) and privacy (via LINDDUN) concerns in a comprehensive manner. This ensured that no critical security or privacy aspects were overlooked. By using both frameworks side by side, we could categorize threats with a higher level of granularity and provide more precise controls for addressing them. We improved the contribution section in the Introduction and the description of the threat category in Section 2.2 to make this clear.
- Besides that, authors should examine results from EU funded projects related to eIDAS as EU has funded plenty of research projects in the respective HORIZON calls.
    - We had actively participated in the risk assessment analysis in the EU-funded project POTENTIAL, one of the large-scale pilots. Our threat model was one of the inputs to the analysis. We added this information in the Introduction section. We cannot provide more details, comparisons, and references as the documents are not publicly available.
- Finally, it is not clear which criteria authors used for selecting the controls, the pairs of threats and vulnerabilities etc?
    - The selection of threats, controls, and vulnerabilities applicable to the EUDIW context was conducted through a comprehensive state-of-the-art analysis, in addition to a thorough analysis with internal discussions. We have added the sources for all of them and described the selection criteria in Section 2.2.
- Why there is no impact assessment on these pairs? Which high-level technical and organisational security and privacy measures should be implemented as the result of this risk analysis for safeguarding eIDAS 2.0?
    - We have decided to focus first on threat modeling (i.e., identification of threats and related vulnerabilities and controls). The impact assessment as part of the risk assessment will be performed in future work. We clarify this by removing the risk from Figure 2 (old Figure 4), highlighting the importance of the threat modeling as a prerequisite of a risk assessment, and adding the risk assessment in future work. In addition, a preliminary discussion in terms of implementation

    - difficulty and effectiveness for some security controls has been added in Section 3.

----------------------- **REVIEW 3** ---------------------

- From an academic viewpoint, however, the novelty of the contribution is not very clear.
    - To better highlight the novelty of this paper, we have revised the introduction and modified the list of contributions.
- The threats reported in Table 1 are very generic and high-level. They read like a list of well-known threats which apply to many systems, superficially tailored to the entities in the DIW context. As a reader, the information gained from Table 1 is minimal. Table 2 is better because threats are much more technical and specific. However, it is not clear how the threats for Tables 1/2 were selected from the much larger set if identified threats. Table 2 lists 5 threats of 13 identified ones. What are the 8 omitted threats, and what were the criteria for omitting them? It may have been better to focus only on specific threats, but comprehensively report all of them.
    - As suggested by the reviewer, we have updated Tables 1 and 2 by bringing the complete list of protocol-specific threats and keeping only a subset of the more general threats (the complete list is available in Appendix C or in our companion website). In addition, we highlight that more information can be found on our companion website.
- Especially for the generic threats, I am missing an answer to the "so what?" question, perhaps in a discussion section.
    - As highlighted in the introduction, the contribution related to the generic threats is the contextualization and mapping with the newly identified assets in the context of the EUDIW. In addition, we discuss some lessons learned in the new Section 3.

Protecting Digital Identity Wallet: A Threat Model in the Age of eIDAS 2.0

Amir Sharif[1], Zahra Ebadi Ansaroudi[1], Giada Sciarretta[1], Daniela Pöhn[3], Majid Mollaeefar[1], Wolfgang Hommel[3], and Silvio Ranise[1,2]

[1] Center for Cybersecurity, FBK Trento, Italy
{asharif, zebadiansaroudi,g.sciarretta,mmollaeefar,ranise}@fbk.eu
[2] Department of Mathematics, University of Trento, Trento, Italy
[3] University of Bundeswehr Munich, Munich, Germany
{daniela.poehn,wolfgang.hommel}@unibw.de

Abstract. The revised eIDAS regulation (eIDAS 2.0) advocates a shift from federated identity management systems (such as SAML and OpenID Connect) to user-centric identity-based systems. It defines the European Digital Identity Wallet as a key component. The main goal is to enhance privacy by empowering citizens to selectively disclose personal data in a controlled way. To facilitate the implementation of an interoperable Wallet solution, the EU Commission published a reference architecture and identified a high-level set of requirements. However, comprehensive security and privacy guidelines to ensure a secure and privacy-preserving solution are still missing. To address this gap, we provide threat modeling explicitly designed for the Digital Identity Wallet context. This allows for identifying potential threats and a set of effective controls to secure the implementations.

Keywords: Digital Identity Wallet · Threat Modeling · eIDAS 2.0.

1 Introduction

In traditional identity management, a central identity provider manages the user's digital identity for all the federated services (relying parties). Thereby, users do not need to register with each relying party, which reduces the number of credentials to remember. This approach that is based on browser-based standard protocols (e.g., SAML [6], OAuth 2.0 [19], and OpenID Connect [47]) is the most widely used by governmental (e.g., electronic IDentification, Authentication and Trust Services (eIDAS) identity schemes [10]) and private solutions (e.g., social login). However, there are potential security and privacy concerns associated with these systems: (*i*) the centralized identity provider may collect user's activity across various relying parties; (*ii*) increased risk of data breaches from storing data in a centralized system; (*iii*) data may be shared with other entities without the user's knowledge; and (*iv*) oversharing of user's personally identifiable information (PII) like birthdate and credit card numbers can be misused by relying parties to track the user's across various relying parties and monetize the user data. A potential way to address these security and privacy concerns is to give the control of the identity data back to the user, which is possible in self-sovereign identity (SSI) based systems [3].

An SSI-based system is user-centric and gives users control over information sharing, allowing them to reveal only the required attributes [36]. Thus, data is stored, managed, and presented by the users rather than the identity providers, mitigating the privacy concerns previously described. These considerations were at the basis of the revised eIDAS regulation [11] (eIDAS 2.0) that advocates for the shift from a traditional to a user-centric-based model and introduces the European Digital Identity Wallet (EUDIW) as the main building block of the future European digital identity infrastructure across Member States. To assist the developers with the implementation of EUDIW, the EU Commission published a draft of the "Architecture and Reference Framework (ARF)" to establish a set of common standards, technical specifications, and best practices to serve as the foundation [13]. However, this document only provides a high-level set of the functional and non-functional requirements that must be satisfied by the implementations during the issuance of the credentials to the EUDIW and their presentation to the relying parties. To elaborate, let us consider the following example: the ARF [13] mentions the OpenID for Verifiable Credential Issuance (OID4VCI) [32] and other optional protocols without explaining the technical details regarding how to securely deploy them within the EUDIW.

In this paper, by analyzing the ARF and relevant academic literature and standards we perform a threat modeling that enables us to (*i*) identify, contextualize and group the potential threats that apply to the EUDIW by specifying threat sources, threat categories (by utilizing STRIDE [35] and LINDDUN [29] frameworks), vulnerabilities, attack scenarios, and the affected assets; and (*ii*) identify a proper set of security and privacy controls that mitigate the identified threats. As a result, we provide the first detailed threat model of EUDIW that collects and contextualizes known threats (e.g., data minimization violation) for the newly identified assets and discusses new threats specific to the EUDIW protocols and components (e.g., unauthorized presentation response forwarding).

Given the uncertainty and ongoing revisions of the technical specifications for the EUDIW infrastructure, we have chosen not to provide a risk assessment that is likely to undergo significant changes in the future. Instead, we have focused our analysis on the more stable features, leaving certain aspects such as authentic source and secure storage out of the scope. While our threat model is not exhaustive, we believe the identified list of assets, threats, and security controls offers valuable insights that can contribute to other threat models and risk assessments within the EUDIW context. Accordingly, as part of European-funded projects within the EUDIW context, we are involved in the POTENTIAL large-scale pilot [14], where our threat model analysis served as an input to the risk assessment process. In addition as members of the W3C Threat Modeling Community Group [59], we plan to further contribute to the threat analysis in the context of decentralized identity systems.

Paper Structure. In Section 2, we present the method we follow to perform threat modeling in the context of EUDIW, which includes a brief overview of the general architecture of the EUDIW, and our threat characterization. In Section 3 we discuss our findings and share the lessons learned. Section 4 contrasts our approach with related works, and finally, we conclude the paper and suggest directions for future research in Section 5.

2 EUDIW Threat Modeling

As a preliminary step to conducting an accurate and effective risk assessment, knowing what potential threats exist is essential. This section presents the method we followed to perform the threat modeling in the context of the EUDIW which consists of the following steps:

Context Establishment is a critical initial step to provide a complete understanding of the system under analysis. This step details the key components of the systems to help identify where the key assets lie and how they interact.

Threat Characterization aims to identify potential threats and attack scenarios applicable to the system, organizing them by threat sources, categories, and the affected assets. It additionally specifies the vulnerabilities, and relevant security and privacy controls that allow to mitigate the identified threats.

2.1 Context Establishment

As a result of the context establishment step, this section outlines the general technical architecture of the EUDIW as shown in Figure 1, highlighting: (1) the EUDIW ecosystem entities, (2) the main interactions that happened in the issuance and presentation of credentials, and (3) the main interfaces involved in the ecosystem for the relevant interactions.

Entities. The main entities involved in the EUDIW infrastructure are:

Wallet Provider is responsible for releasing a Wallet to a Holder that is the user of the Wallet.

Wallet is the mobile application that is installed on the Holder's device and provides graphical interfaces for user interaction with the Verifier, Issuers, and Wallet Provider;

Trust Anchor represents an entity that forms the trust infrastructure for the validation of certificates, raw public keys, and entity metadata as compliant with a pre-established trust framework;

Issuer represents the Issuer of credentials, which can be Personal Identification Data (PID) and (Qualified) Electronic Attestation of Attributes ((Q)EAAs);

Verifier represents a service that implements an authentication system requiring electronic attribute attestation submissions as an authentication mechanism before providing a resource to the Holder.

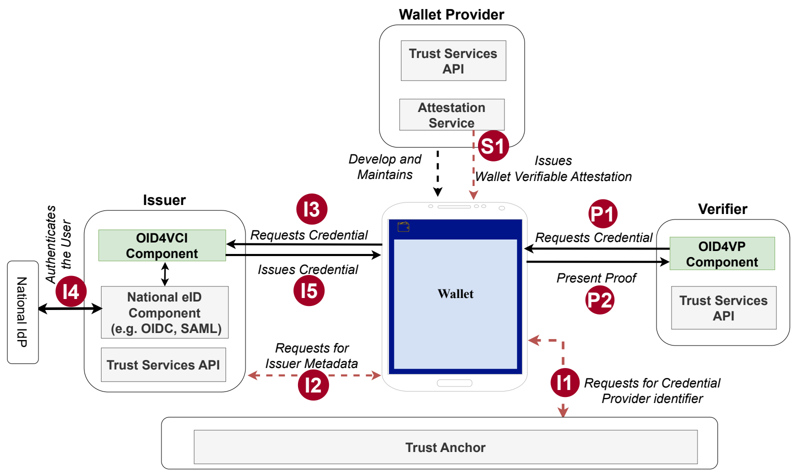

Fig. 1. General architecture and high-level flows with the relevant interfaces involved in the issuance and presentation flows.

Phases. To ensure a secure and privacy-preserving functionality of EUDIW, the first time that the Wallet is started, a **Wallet Instance Setup (S1)** occurs. This step is either done individually or together with the issuance of the PID credential for the EUDIW and includes the release of the Wallet Attestation issued by the Attestation Service of the Wallet Provider, which asserts the genuineness and the compliance of the Wallet with the trust framework. If the Wallet Attestation is available or issued, the Wallet can proceed with the following steps for the issuance or presentation of its credential for its intended use case:

- **I1. Issuer Discovery** enables the Wallet to discover a list of trusted Issuers by querying the Trust Service API provided by the Trust Anchor;
- **I2. Credential Issuer Metadata** enables the Wallet to establish trust with the Issuer according to the trust framework and to obtain the Metadata that discloses the formats of the credential, the algorithms supported, and any other parameter required for interoperability needs;
- **I3-5. Credential Issuance** are the steps in which the Wallet sends a request for the credential to the Issuer, and the Issuer, after successful Holder authentication, will release the requested credential to the Wallet;
- **P1. Credential Presentation Request** sent by the Verifier when the Wallet wants to access a service of the Verifier.
- **P2. Credential Presentation Response** provides the Holder's credential that best matches the Verifier request and access to the requested service.

Interfaces. The interfaces or components essential for the EUDIW functionality in credential issuance and presentation include:

Attestation Service is in charge of issuing Wallet Attestation to the Wallet. The Wallet Attestation certifies the genuineness and authenticity of the Wallet and its compliance with the security and privacy requirements.

Trust Service API provides a way for the Issuer/Wallet Provider/Verifier to communicate with the Trust Anchor and obtain trust-related information;

OID4VCI Component is based on the "OpenID for Verifiable Credential Issuance" specification [32] to release credentials;

National eID Component authenticates the Holder with the National Digital Identity Providers, based on OpenID Connect Core 1.0 [47] or SAML2 [6];

OID4VP Component is based on the "OpenID for Verifiable Presentations" specification [54] to request for presentation of credentials.

2.2 Threat Characterization

Our approach to threat characterization is based on a general threat analysis ontology, illustrated in Figure 2, which draws upon established security ontologies in the literature, such as those developed by the National Institute of Standards and Technology (NIST) for enterprise-level risk assessment [52]. To narrow down our threat analysis, we currently consider the main EUDIW entities (Issuers, Wallet Providers, Wallets, Holders, and Verifiers) and phases (issuance and presentation), while considering out of scope: (*i*) other EUDIW entities (e.g., authentic source and qualified trust service provider), (*ii*) threats specific to secure storage of keys based on various technologies (e.g., Trusted Execution Environment, Hardware Security Modules, etc.), (*iii*) various deployment of trust models (e.g., OpenID Federation [22]), and (*iv*) stand-alone implementation of the entities (e.g., their software security). In addition, we decided to consider protocols used in the SSI context (such as DIDComm) out of scope, and only focus on the OpenID technological stack as it is explicitly mentioned in the ARF [13], EUDIW Reference Implementation [12], German Wallet Architecture Proposal [5], Italian Wallet Architecture Proposal [23] and Swiss E-ID & Trust Infrastructure [53].

Table 1 and Table 2 show a subset of the threats identified in our analysis (see Appendix C or our companion website [48] for the complete list). For each threat, we have specified five entries, where:

- **Threat Categories (TC)** entry assigns each threat to a category based on the STRIDE [35] and LINDDUN [29] frameworks. STRIDE, developed by Microsoft, categorizes security threats into Spoofing (SP), Tampering (TA), Repudiation (RE), Information Disclosure (ID), Denial of Service (DS), and Elevation of Privileges (EP). Meanwhile, LINDDUN, a privacy-focused framework, categorizes privacy threats into Linkability (LN), Identifiability (IF), Non-Repudiation (NR), Detectability (DT), Data Disclosure (DD), Unawareness/Unintervenability (UU), and Non-Compliance (NC). These categories are particularly well-suited to digital identity ecosystems, where issues like identity spoofing and data tampering (STRIDE), as well as linkability and identifiability (LINDDUN), are critical. These frameworks are widely

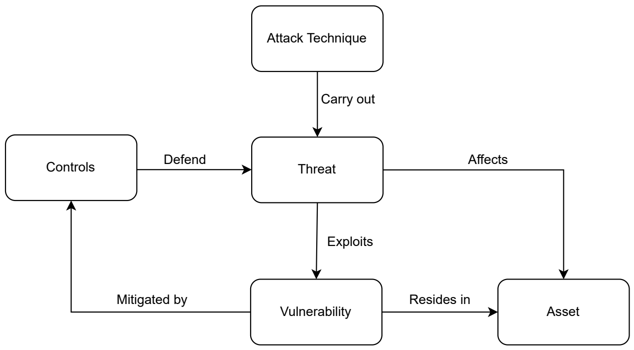

Fig. 2. Threat analysis ontology inspired from the NIST [52].

recognized within the research community [1,4,26,44,45], ensuring that our approach follows best practices. Additionally, using both frameworks allowed us to categorize threats with greater granularity, enhancing our ability to identify and mitigate critical threats while ensuring no security or privacy aspect is overlooked.

- **Attack Techniques (AT)** entry reports the applicable methods or strategies that an attacker can use to carry out the identified threat. We have categorized similar attack techniques under a unified category name based on shared characteristics and methods employed by the AT. For example, *AT15* repudiation category includes all the identified ATs (e.g., *AT15.1* describing a malicious Holder that could deny the request/gain of credentials by trying to remove the logs, in *T6* repudiation of requested/obtained/revoked verifiable credentials / Wallet trust evidence) that are based on intercepting communications between involved entities without their knowledge;
- **Vulnerabilities (V)** entry reports the inherent weaknesses or flaws that can be exploited by the identified threat. For example, *V12* insecure key management that arises from inadequate protection and management of authentication credentials, such as private keys, can be exploited by *T6*;
- **Affected Entity/Assets (AA)** entry reports the affected data, service, or interaction of an entity by the identified threat. The entities' identified assets are denoted by a combined identifier, including two parts separated by "-", where the first part shows the entity within the system and the latter is for the asset type based on the following notations:
  - Entities: Issuer (I), Verifier (V), Holder (H), Wallet Provider (WP), and Wallet (W);
  - Asset types: Data (DA), Service (SA), and Interaction (IA).

  For example, the code IV-SA stands for Issuer and Verifier Services asset;
- **Controls (C)** entry reports the security or privacy measures to mitigate the identified threats. For example, *C18* secure identification of Holders can mitigate *T7* credential theft.

This selection of threats was conducted through a comprehensive state-of-the-art analysis. We reviewed and analyzed threats identified for identity Wallets, SSI systems, and analogous applications such as cryptocurrency wallets within the literature. This was complemented by an examination of relevant standards, specifications, and the ARF. By synthesizing these sources, we identified and defined the threats most pertinent to the EUDIW, ensuring that our threat model is both comprehensive and context-specific.

Following the identification of threats, we conducted a detailed analysis of the corresponding elements, including vulnerabilities and controls. The vulnerability analysis was guided by established best practices, such as those from the World Wide Web Consortium (W3C) and the OpenID Foundation, and well-known vulnerability databases like the Open Worldwide Application Security Project (OWASP) and the Common Weakness Enumeration (CWE). This ensured that the vulnerabilities were accurately represented, aligning with the security challenges inherent in the EUDIW ecosystem. To mitigate these threats and vulnerabilities, controls were carefully selected based on their proven effectiveness and alignment with recognized security standards, including those from W3C, OpenID Foundation, and other industry best practices. Additionally, we referred to the ARF requirements [13], identifying controls that addressed both architectural and operational aspects of the EUDIW.

Table 1: An excerpt of General Threats in the EUDIW ecosystem.

T6. Repudiation of requested/obtained/revoked verifiable credentials / Wallet Trust Evidence involves an entity denying the initiation or receipt of a transaction or an action [18,45].	
TC:	RE
AT:	(AT15) repudiation (e.g., in AT15.1, a malicious Holder can deny the request/gain of credentials by trying to remove the logs)
V:	log tampering / inappropriate logging control (V11), insecure key management (V12), abuse of functionality (V13), and data integrity (V17)
AA:	IVWPW-SDA
C:	secure logging mechanism (C1), secure storage (C4), Holder revocation of credentials (C11), notification informing the Holder of any credential request/obtained (C13)
T7. Credential theft refers to the malicious acquisition of the Holder's credentials to present it to the Verifier and obtain unauthorized access to the service [8].	
TC:	SP
AT:	(AT2) value/parameter injection (e.g., AT2.4 exploits the flaw in the presentation request to inject a stolen/leaked credential in the communication with the Verifier to access the service)
V:	injection (V14) or insecure exchange protocol (V21)
AA:	VH-DA
C:	secure communication (C18), the validation of request/response messages (C23), and key binding (C32)
T15. Exploitation of faulty credential status check mechanism occurs when the credential validity check mechanism is not designed in a privacy-preserving way (e.g., it demands direct communication between the Verifier and Issuer). Thus, it would leak sensitive information about the usage of credentials and enable Holder's tracking [15].	
TC:	LN
AT:	(AT20) credential validity technique (e.g., a curious Issuer can use the flaw in the Verifier's credential validity check by the Verifier to obtain information about the Wallet's actions and perform the Holder's activity linking)
V:	improper credential verification (V18) and inadequate anonymization (V19)
AA:	H-DA
C:	privacy by design (C26)

T16. Unauthorized presentation response (VP token) forwarding aims to impersonate the Holder at the Verifier by forwarding the VP token extracted from the interaction of Wallet and Verifier into the attacker session with the Verifier [15].

TC:	SP, EP
AT:	(AT21) presentation reply (e.g., an attacker can send the stolen presentation to the Verifier instead of a presentation created by his Wallet)
V:	improper credential verification (V18), insufficient session management (V20), insecure exchange protocol (V21)
AA:	V-SA, WV-IA
C:	session binding (C24), `aud` parameter in the presentation response (C28)

T18. Registration on non-compliant entity aims to register a non-compliant entity to operate in the ecosystem for a malicious purpose.

TC:	NC
AT:	(AT22) certification scheme bypass (e.g., an attacker can misuse the flaw within a certification system to operate a non-compliant entity
V:	improper certification validation (V24)
AA:	IVWP-DSIA
C:	cybersecurity certification auditing (C29)

T19. Exploitation of faulty data exchange protocol implementation advantages the improper implementations of the exchange protocols without following the best practices that can lead to increasing the attack surface.

TC:	NC, SP, ID, IF
AT:	(AT23) token theft (e.g., obtains the Holder `vp_token` by exploiting a flaw on presenting the access token)
V:	configuration (V7), insecure data exchange protocol (V21)
AA:	H-DA
C:	secure data exchange protocol implementation (C30)

T30. Data minimization violation refers to the practice of requesting, obtaining, and storing more personal information from individuals than is strictly necessary for the intended service or transaction [18,25,34].

TC:	DD, NC, ID
AT:	(AT29) information disclosure attacks (e.g., Verifiers may request extensive or unnecessary attributes, revealing Holder's information)
V:	excessive data exposure (V29)
AA:	H-DA
C:	Privacy by design (C26)

Table 2: A preliminary list of OID4VCI/OID4VP Threats in the EUDIW ecosystem.

T1. Acceptance of wildcard as redirect URI is intended to misuse the Issuer redirect URI registration system with the aim of sending the Holder's user agent to a place under the attacker's control [31].

TC:	SP, ID, IF
AT:	(AT35) Redirect mechanism registration attack (e.g., the attacker can exploit the flaws in the registration of a wild card as a redirect URI to return the authorization response to a URL under attacker control to obtain the authorization code, exchange it for an Access Token, and consequently obtain Holder's credential)
VT:	injection (V14), lack of exact redirect URI matching (V32)
AA:	H-DA, I-SA
C:	exact redirect URI matching (C33)

T2. Use of stolen/leaked push authorization `request_uri` parameter involves the use of the stolen/leaked push authorization response `request_uri` parameter to obtain the authorization code and use it for malicious purposes [30].

TC:	SP, EP
AT:	(AT2) value/parameter Injection (e.g., in AT2.8, an attacker can try to inject/re-use the leaked intercepted `request_uri` parameter into its session with an Issuer to maliciously obtain an authorization code that later can be exchanged it for an Access token which consequently enables the attacker to obtain credentials)
V:	injection (V14), improper protocol parameters verification (V22)
AA:	WH-DA
C:	short and one-time use `request_uri` (C34)

T3. Insufficient entropy of `request_uri` parameter involves the threat actor trying to guess the value of the `request_uri` parameter that enables the threat actor to impersonate the Wallet [30].
TC: SP, EP
AT: (AT36) request URI object guessing (e.g., an attacker could attempt to guess and replay a valid request URI value and try to impersonate the respective Wallet, for that specific request)
V: lack of sufficient entropy for `request_uri` (V33)
AA: H-DA
C: `request_uri` value randomness (C35)

T4. Use of stolen/leaked one-time use Wallet control proof involves the use of stolen/leaked Wallet control proof for malicious purposes [8].
TC: SP
AT: (AT2) value/parameter Injection (e.g., in AT2.9, an attacker can try to inject/re-use the leaked/stolen Wallet control proof to obtain the Holder's credentials)
V: injection (V14), improper protocol parameters verification (V22)
AA: H-DA
C: server-provided nonce (C36)

T5. Wallet control proof modification involves the unauthorized modification of the wallet control proof within the credential request to the values that are known by the attacker [8].
TC: TA, SP
AT: (AT2) value/parameter Injection (e.g., in AT2.10, an attacker can intercept the communication between the Wallet and Issuer to maliciously modify the Wallet control proof with a proof that is created by the attacker)
V: injection (V14), improper protocol parameters verification (V22)
AA: H-DA
C: use of sender-constrained access token (C37), Authorization request validation (C39)

T6. Use of stolen/leaked authorization code involves the threat actor trying to steal the authorization code or re-use the leaked authorization code to exchange it for an access token to gain unauthorized access to Holder's resources [32,31].
TC: SP, EP
AT: (AT2) value/parameter injection (e.g., in AT2.11, an attacker can use a man-in-the-middle attack to intercept the communication between the Wallet and Issuer to steal the code and try to inject it into his session with an Issuer)
(AT29) information disclosure attacks (e.g., the contents of the authorization request URI or the authorization response URI can unintentionally be disclosed to attackers through the Referer HTTP header if it is not set properly and enables the attacker to access the credentials such as the authorization code.)
(AT37) Mix-Up (e.g., an attacker can launch a Mix-Up attack to obtain protocol-sensitive values such as authorization code and misuse it)
V: injection (V14), improper protocol parameters verification (V22)
AA: H-DA
C: using `iss` parameter (C49), use of PKCE (Proof Key for Code Exchange) (C51)

T7. Access Token Theft involves the threat actor trying to steal or use the leaked access token to exchange it to obtain the Holder's credential [32,31].
TC: SP, EP
AT: (AT1) man-in-the-middle (e.g., in AT1.1, an attacker can intercept messages between the Issuer and Wallet to steal sensitive data such as code, tokens, personal data, or credentials)
(AT37) Mix-Up (e.g., an attacker can launch a Mix-Up attack to obtain protocol-sensitive values such as authorization code and misuse it)
V: injection (V14)
AA: H-DA
C: use of sender-constrained access token (C37), using `iss` parameter (C49)

T8. Session misuse involves the threat actor trying to inject a request to the redirect URI of the legitimate Wallet/Verifier on the Holder's user agent to cause the Holder to access resources under the attacker's control [31].
TC: EP, SP
AT: (AT8) CSRF (e.g., an attacker can perform a CSRF attack to redirect the Holder to a `response_uri` endpoint that is under the attacker's control and consequently deceive the Holder into sharing the credential with a Verifier under the attacker's control)
V: improper protocol parameters verification (V22)
AA: WV-IA, H-DA
C: use of state/nonce (C38), use of proof key for code exchange (C51)

T9. Exploit insecure Wallet invocation involves the threat actor trying to register the same redirect scheme as the legit Wallet for a malicious purpose [32].

TC: SP, ID, IF
AT: (AT45) same Custom URI attack (e.g., an attacker can register the same Custom URI scheme as the same legit Wallet, as multiple Wallet can typically register the same Custom URI scheme that makes it indeterminate for OS to which Wallet it should redirect. Thus, the attacker can obtain sensitive information)
V: lack of secure Wallet invocation (V34)
AA: W-DSIA
C: using universal app links (C50)

T10. Acceptance of faulty authorization/token/credential/wallet trust evidence requests is intended either to modify the values within the authorization/token/credential/wallet trust evidence request to the values under attacker control or to remove some parameters for malicious purposes [32].
TC: SP, TA
AT: (AT1) man-in-the-middle (e.g., An attacker can use man-in-the-middle type of attacks to modify the authorization request values to values that are known to the attacker to bypass the checks at the Issuer. For example, it can modify the PKCE values to a value that is known to the attacker to bypass the PKCE check)
(AT38) PKCE Downgrade attack (e.g., An attacker can perform a PKCE downgrade attack that enables an attacker to inject an authorization code (and with that, an access token) that is bound to the attacker's resources into a session between their victim (Holder) and the Wallet)
(AT39) redirection attack (e.g., an attacker can misuse the lack of proper redirect URI validation in the authorization request to modify it to an address under attacker control to steal protocol-sensitive data (code, access token) and consequently access Holder's credential)
V: improper protocol parameters verification (V22)
AA: WH-DA
C: authorization request validation (C39), token request validation (C40), credential request validation (C41), Wallet trust evidence request validation (C42)

T11. Acceptance of faulty presentation request occurs when the Wallet fails to sufficiently verify the presentation request that is received from Verifier [54].
TC: SP, ID, IF
AT: (AT40) fake credential presentation request (e.g., an attacker (malicious Verifier) can exploit the improper checks by the Wallet to send a presentation request to the Wallet pretending that it is playing the role of Legit Verifier)
V: improper protocol parameters verification (V22)
AA: H-DA
C: presentation request validation (C43)

T12. Acceptance of faulty presentation response occurs when the Verifier fails to sufficiently verify the presentation response that is received from the Wallet [54].
TC: EP, ID, IF
AT: (AT17) credential modification (e.g., an attacker can misuse the lack of proper redirect URI validation in the authorization request to modify it to an address under attacker control to steal protocol-sensitive data (code, access token) and consequently access Holder's credential)
(AT41) fake credential presentation response attack (e.g., an attacker can exploit the improper checks by the Verifier to send a presentation response to the Verifier from another session to access the Verifier service)
V: improper protocol parameters verification (V22)
AA: V-SA
C: presentation response validation (C44)

T13. Acceptance of faulty authorization/token/credential/wallet trust evidence response occurs when the Wallet fails to adequately verify the authorization, token, credential, and wallet trust evidence responses received from the Issuer/Wallet Provider [32].
TC: SP, EP
AT: (AT1) man-in-the-middle (e.g., an attacker (Malicious entity or Malicious Wallet Provider) can use man-in-the-middle type of attacks to modify the CNF parameter in the Wallet Trust Evidence (WTE) request to a value under the attacker's control to issue a WTE that is bound to the attacker's public key. In this way, the attacker can misuse the WTE later on in the protocol to bypass the checks)
(AT42) fake credential issuance attack (e.g., an attacker can exploit this threat to intervene with the issuance process and issue fake credentials from an Issuer under attacker control to the Holder)
(AT43) response swapping (e.g., an attacker can substitute the response from its session to the Holder's session to bind the Holder's session to the attacker's session)

	(AT44) `request_uri` swapping (e.g., an attacker in the case of using PAR can substitute the PAR response (`request_uri`) using a swapping attack that enables the attacker to capture the `request_uri` from one request and then substitute it into a different authorization request)
V:	improper protocol parameters verification (V22)
AA:	H-DA
C:	authorization response validation (C45), token response validation (C46), credential response validation (C47), Wallet trust evidence response validation (C48), using `iss` parameter (C49)

3 Discussion

We have identified a total of **49** threats, of which **13** are specific to OID4VCI [32] and OID4VP [54], **36** are more general and consider various phases of the Wallet's development and use. Due to space constraints, we report a list of **7** out of the **36** in Table 1 (where all are listed in Appendix C) and provide the complete list of the **13** protocol-specific threats in Table 2.

From our analysis, we have collected the following lessons learned:

– Holder (28 out of 49), Verifier (21 out of 49), and Issuer (16 out of 49) are the most targeted entities.
– Spoofing emerged as the most common threat in both our general (19 out of 36) and the OID4VCI/OID4VP-specific threat analysis (12 out of 13). The impact of spoofing is particularly severe, ranging from the issuance of fake credentials to identity theft, posing significant risks to the EUDIW ecosystem. Additionally, information disclosure (12 out of 36) and elevation of privilege (7 out of 13) are the next prevalent threats in general and OID4VCI/OID4VP context, respectively. The former potentially allows attackers to access and misuse sensitive personal information, including PII and the Holder's private keys, while the latter can lead to unauthorized access to the resources or services granted to the Holder;
– Identifying (14 out of 49) and Non-compliance (5 out of 49) are the most prevalent privacy threats within our analysis. Identifying threats poses a significant risk to user privacy, as they enable attackers to uncover the Holder's identity and carry out profiling activities. Non-compliance, on the other hand, is a critical threat to avoid, as failure to adhere to regulations could result in hefty fines from regulatory authorities;
– Threat exploitation can occur at various stages within the ecosystem, presenting multiple opportunities for attackers. Of the identified threats, 18 affect the ecosystem as a whole, while 13 specifically target the issuance phase and 17 focus on the presentation phase. Additionally, two threats are related to the setup process, and one targets the trust establishment. This distribution underscores the wide range of attack surfaces and highlights the critical need for robust security and privacy measures throughout all phases;
– Using `iss` parameter (*C49*), privacy by design (*C26*), Entity training, education and awareness (*C16*), secure storage (*C4*), general platform security (*C19*) and Holder revocation of credentials (*C11*) are among the top security and privacy controls within our analysis. *C49*, *C26*, and *C16* significantly

help with providing security and privacy under the protocol hood. *C49* can easily be implemented as it needs to be added to the OID4VC implementation. It can effectively mitigate the interference of an attacker within the credential issuance (see T6-7, and T13 in Table 2), which can lead to unauthorized access to Holder's resources or issuance of fake credentials. *C26* due to the involvement of cryptographic techniques (e.g., zero-knowledge proofs) is a more demanding implementation task, however, it is an effective mitigation against threats such as data minimization (T8) and linkability (T36). *C16* can help with assuring that Holders and Issuers/Verifiers operators are sufficiently educated to reduce accidental security problems through social engineering or phishing campaigns that may put the whole ecosystem at risk. The integration of this control is more demanding due to the need for comprehensive and continuous training, however, it may be less effective. *C4*, *C11*, and *C19* can be categorized under the hood of Wallet security. The first two contribute to the Wallet hardware and Wallet software code security, while the latter can be seen as a kill switch that enables the Holders to revoke their credentials in case of the misuse of their credentials. *C4* demands moderate implementation resources and is limited to the capabilities that are provided by the device hardware (especially in a local solution). However, it can be an effective mitigation against threats such as theft of sensitive information. *C11*, while difficult to implement due to its cryptographic complexity, provides a moderate level of protection against credential theft. This level of protection is because the time lag between detecting an attack and revoking the compromised credentials creates a window of opportunity for attackers to exploit. Lastly, *C19* is easier to implement as it is provided by the mobile OS and accessible through APIs. If it is implemented properly, it can be an effective mitigation against threats such as Wallet compromise.

Interested readers can find further details in the appendix – vulnerabilities in Appendix A, controls in Appendix B, and the complete list of threats in Appendix C – and the definitions of the threat categories, assets, vulnerabilities, attack techniques, and controls in the companion website [48].

4 Related Work

This section reviews key research on threat modeling for digital identity wallets, emphasizing recent developments. In 2019, Bisztray et al. compared three Data Protection Impact Assessment (DPIA) methods—LINDDUN, CNIL, and ISO/IEC 29134:2017—evaluating their practical application and GDPR compliance [4]. LINDDUN, while intuitive, lacked a thorough risk assessment and detailed GDPR compliance integration [57]. These studies advocate for advancements in threat modeling and DPIA methodologies for robust privacy protections. In 2021, Gon Kim et al. used STRIDE threat modeling to analyze decentralized identifiers (DID), identifying vulnerabilities and countermeasures across several domains, emphasizing a holistic security approach in DID development [26]. In 2022, Pellegrino and Sainio introduced KRAKEN, a privacy-centric data exchange platform utilizing multi-party computation, SSI, and blockchain, confirmed to be effective against unauthorized access and privacy challenges through LINDDUN analysis [44]. Ahmadjee et al. explored the security impact of blockchain architectural decisions, offering a STRIDE-based taxonomy for secure blockchain application development [1]. Pöhn et al. applied STRIDE to threat modeling in SSI, identifying threats and mitigations within the SSI ecosystem [45]. Grüner et al. combined STRIDE and CVSSv3 to reveal a higher threat surface in SSI systems [18]. Kersic et al. unified on-chain and off-chain SSI management using STRIDE-based threat modeling to enhance Wallet security and privacy [25]. Torongo et al. introduced the BDIMHS framework using Hyperledger technologies to improve healthcare identity management, focusing on patient data control and privacy with identity Wallets [55]. These studies typically define high-level requirements or threats for specific scenarios or domains, such as blockchain-based cryptocurrency or identity Wallets [25,55,57], or list threats and attacks applicable to SSI solutions [7,27,37,18]. In contrast, our approach adopts a comprehensive threat modeling strategy using STRIDE and LINDDUN frameworks, considering key entities—Issuers, Wallet Providers, Wallets, Holders and Verifiers—and providing controls to ensure the security and privacy of the EUDIW implementation.

5 Conclusion

The drafts and recommendations on the EUDIW, such as the ARF, are rather generic, without details on how to implement the EUDIW securely. To fill this gap, we outlined our systematic threat modeling method using STRIDE and LINDDUN frameworks. Our threat model as a reference tool offers valuable insights and practical guidance for the entities within the EUDIW ecosystem. It helps these entities easily identify specific threats they may face and apply relevant mitigations, making the findings actionable.

In the future, we first intend to broaden our preliminary list of identified threats for OID4VCI and OID4VP protocols to make it more comprehensive. Second, we plan to perform a security analysis on the available EUDIW solutions in the wild to (i) evaluate their implementation against the identified threats and how their implementation mitigates them and (ii) assess their compliance with the eIDAS [11] regulation. Finally, we plan to complement our threat modeling with a control-based risk assessment to highlight how the risk level of EUDIW is changing based on the implementation of controls within the solution.

References

1. Sabreen Ahmadjee and et al. A study on blockchain architecture design decisions and their security attacks and threats. ACM Transactions on Software Engineering and Methodology (TOSEM), 2022.
2. Md Rayhan Ahmed and et al. Blockchain-based identity management system and self-sovereign identity ecosystem: A comprehensive survey. IEEE Access, 2022.

3. Zahra Ebadi Ansaroudi and et al. Control is nothing without trust a first look into digital identity wallet trends. In IFIP Annual Conference on Data and Applications Security and Privacy. Springer, 2023.
4. Tamas Bisztray and Nils Gruschka. Privacy impact assessment: comparing methodologies with a focus on practicality. In Nordic Conference on Secure IT Systems. Springer, 2019.
5. Bundesministeriums des Innern und für Heimat. German Wallet Architecture Proposal. https://gitlab.opencode.de/bmi/eudi-wallet/eidas-2.0-architekturkonzept/-/tree/main, 2024.
6. Scott Cantor. SAML Version 2.0 Errata 05, 2012.
7. P Dingle and et al. Alice attempts to abuse a verifiable credential, 2023.
8. ENISA. Digital Identity: Leveraging the SSI Concept to Build Trust. https://www.enisa.europa.eu/publications/digital-identity-leveraging-the-ssi-concept-to-build-trust, 2022.
9. Christina Erler and et al. Threat modeling to design a decentralized health data management application. In International Conference on Information Technology & Systems, 2023.
10. EU. eIDAS Regulation (EU) 910/2014. https://eur-lex.europa.eu/eli/reg/2014/910/oj, 2014.
11. EU. eIDAS Regulation (Eu) 2024/1183. https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32024R1183, 2024.
12. EU. EUDI Wallet Reference Implementation. https://github.com/eu-digital-identity-wallet/.github/blob/main/profile/reference-implementation.md, 2024.
13. EU. The European Digital Identity Wallet. https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/1.4.0/, 2024.
14. FBK. Pilots for european digital identity wallet. https://st.fbk.eu/projects/POTENTIAL.
15. Daniel Fett and et al. Security and Trust in OpenID4VC Ecosystems. https://openid.github.io/OpenID4VC_SecTrust/draft-oid4vc-security-and-trust.html, 2024.
16. GDPR Advisor. Gdpr and consent management. https://www.gdpr-advisor.com/gdpr-and-consent-management-strategies-for-obtaining-and-managing-consent/, 2023.
17. Google. Overview of the google play integrity api. https://developer.android.com/google/play/integrity/overview, 2024.
18. Andreas Grüner and et al. Analyzing and comparing the security of self-sovereign identity management systems through threat modeling. International Journal of Information Security, 2023.
19. Dick Hardt. The OAuth 2.0 Authorization Framework (RFC6749). Internet Engineering Task Force (IETF), 2012.
20. Fabian Hauck. Openid4vc: formal security analysis using the web infrastructure model. Master's thesis, University of Stuttgart, 2023.
21. Daojing He and et al. Security analysis of cryptocurrency wallets in android-based applications. IEEE Network, 2020.
22. Roland Hedberg and et al. OpenID Federation 1.0 - draft 36. https://openid.net/specs/openid-federation-1_0.html#name-authors-addresses, 2024.
23. IT Wallet. EUDI Wallet Technical Specifications. https://github.com/italia/eudi-wallet-it-docs, 2024.

24. Saqib A Kakvi and et al. Sok: anonymous credentials. In International Conference on Research in Security Standardisation. Springer, 2023.
25. Vid Kersic and et al. Orchestrating digital wallets for on-and off-chain decentralized identity management. IEEE Access, 2023.
26. Bong Gon Kim and et al. A security analysis of blockchain-based did services. IEEE Access, 2021.
27. Anhtuan Le, , and et al. A comparative cyber risk analysis between federated and self-sovereign identity management systems. Data & Policy, 2023.
28. Cong Li and et al. Android-based cryptocurrency wallets: Attacks and countermeasures. In 2020 IEEE International Conference on Blockchain (Blockchain).
29. LINDDUN. LINDDUN Privacy Threat Modeling Framework. https://linddun.org/.
30. Torsten Lodderstedt and et al. OAuth 2.0 Pushed Authorization Requests (RFC9126). Internet Engineering Task Force (IETF), 2021.
31. Torsten Lodderstedt and et al. OAuth 2.0 Security Best Current Practice (draft-ietf-oauth-security-topics-25). Internet Engineering Task Force (IETF), 2024.
32. Torsten Lodderstedt and et al. OpenID4VCI. https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html, 2024.
33. Torsten Lodderstedt and Kristina Yasuda. OpenID4VC High Assurance Interoperability Profile with SD-JWT VC - draft 01. https://github.com/openid/oid4vc-haip-sd-jwt-vc, 2024.
34. Rene Mayrhofer and et al. Towards Threat Modeling for Private Digital Authentication in the Physical World. https://www.digidow.eu/publications/2021-mayrhofer-tr-digidowthreatmodeling/Mayrhofer_2021_DigidowThreatModeling.pdf, 2021.
35. Microsoft. STRIDE Threat Modeling Framework. https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool.
36. Alexander Mühle and et al. A survey on essential components of a self-sovereign identity. Computer Science Review, 2018.
37. Nitin Naik and et al. An attack tree based risk analysis method for investigating attacks and facilitating their mitigations in self-sovereign identity. In 2021 IEEE Symposium Series on Computational Intelligence (SSCI), 2021.
38. NIST. Guide to computer security log management. Technical report, National Institute of Standards and Technology (NIST), 2006.
39. OWASP. Owasp mobile top 10 2024. https://owasp.org/www-project-mobile-top-10/2023-risks/.
40. OWASP. Top 10 web application security risks for 2021. https://owasp.org/Top10/.
41. OWASP. Api3:2019 excessive data exposure. https://owasp.org/API-Security/editions/2019/en/0xa3-excessive-data-exposure/, 2019.
42. OWASP. Mobile application security cheat sheet. https://cheatsheetseries.owasp.org/cheatsheets/Mobile_Application_Security_Cheat_Sheet.html, 2023.
43. OWASP AppSec Research. Improving the security of session management in web applications. https://wiki.owasp.org/images/0/0f/Improving_the_Security_of_Session_Management_in_Web_Applications_-_Philippe_De_Ryck.pdf, 2013.
44. Donato Pellegrino and et al. Architecture for privacy-preserving brokerage of analytics using multi party computation, self sovereign identity and blockchain. University of Turku, 2022.

45. Daniela Pöhn and et al. Modeling the threats to self-sovereign identities. Gesellschaft für Informatik eV, 2023.
46. Matteo Rizzi and et al. Tlsassistant v2: A modular and extensible framework for securing tls. In Proceedings of the 27th ACM on Symposium on Access Control Models and Technologies, 2022.
47. Nat Sakimura and et al. OpenID Connect Core 1.0 incorporating errata set 1. The OpenID Foundation, specification, 335, 2014.
48. Amir Sharif. Protecting digital identity wallet: A threat model in the age of eidas 2.0. https://sites.google.com/view/eu-digital-identity-wallet/home, 2024.
49. Amir Sharif and et al. Best current practices for oauth/oidc native apps: A study of their adoption in popular providers and top-ranked android clients. Journal of Information Security and Applications, 2022.
50. Amir Sharif and et al. Cross-domain sharing of user claims: A design proposal for openid connect attribute authorities. In Proceedings of the 18th International Conference on Availability, Reliability and Security, 2023.
51. Al Tariq Sheik and et al. A comparative study of cyber threats on evolving digital identity systems. In Competitive Advantage in the Digital Economy (CADE 2021). IET, 2021.
52. Anoop Singhal, Samuel Singapogu, et al. Security ontologies for modeling enterprise level risk assessment. In Proceedings of the 2012 Annual Computer Security Applications Conference, Orlando, FL, USA, 2012.
53. SWISSCOMM. Swiss E-ID and Trust Infrastructure. https://github.com/e-id-admin/open-source-community, 2024.
54. Oliver Terbu and et al. OpenID4VCP. https://openid.github.io/OpenID4VP/openid-4-verifiable-presentations-wg-draft.html, 2024.
55. Arnaf Aziz Torongo and Mohsen Toorani. Blockchain-based decentralized identity management for healthcare systems. arXiv preprint arXiv:2307.16239, 2023.
56. Md Shahab Uddin and et al. Horus: A security assessment framework for android crypto wallets. In Security and Privacy in Communication Networks: 17th EAI International Conference, SecureComm 2021.
57. Fatbardh Veseli and et al. Engineering privacy by design: lessons from the design and implementation of an identity wallet platform. In Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing, 2019.
58. Pim Vullers and Gergely Alpár. Efficient selective disclosure on smart cards using idemix. In IFIP Working Conference on Policies and Research in Identity Management. Springer, 2013.
59. W3C. Threat modeling community group. https://www.w3.org/community/tmcg/.
60. W3C. Verifiable credential data integrity 1.0. Technical report, 2023.
61. W3C. Verifiable credentials api v0.3. Technical report, 2023.
62. W3C. Securing verifiable credentials using jose and cose. https://www.w3.org/TR/securing-verifiable-credentials, 2024.
63. W3C. Verifiable credentials data model v2.0. https://www.w3.org/TR/vc-data-model-2.0/, 2024.

A Vulnerabilities

In the following, we provide the definition of Vulnerabilities that are presented within Table 1 and Table 2.

Table 3: Vulnerabilities

V7. **Configuration vulnerabilities** [39]	It addresses weak configurations or misconfigurations. In the former, security measures are not defined, implemented, or maintained properly. In the latter, security settings are deployed with errors.
V11. **Inappropriate logging mechanism** [40]	It occurs when a system's logging capabilities are compromised, allowing an attacker to alter or delete logs to hide their activities.
V12. **Insecure key management** [39]	It arises from inadequate protection and management of authentication credentials, such as private keys.
V13. **Lack of non-repudiation** [51]	This vulnerability occurs when a system lacks mechanisms to prevent entities from denying actions they have performed, such as issuing credentials. For instance, a Wallet Provider might deny issuing Wallet Trust Evidence, depriving users of necessary services without a way to prove otherwise.
V14. **Injection vulnerabilities** [40,39,6	It is a class of security weaknesses that occur when an attacker can supply or inject untrusted data into a system in a way that the data is interpreted and executed as part of a command or query to perform unintended actions.
V17. **Data integrity vulnerability** [40,60]	It occurs when a system lacks effective measures to prevent unauthorized modifications to data. It encompasses the risk of data being altered, tampered with, or corrupted without permission, compromising the accuracy and trustworthiness of the data across its lifecycle.
V18. **Improper credential verification** [62,61]	It occurs when a system fails to adequately verify the credentials or claims (assertions about the identity, attributes, or rights of a user) embedded within credentials, leading to the acceptance of false, manipulated, outdated, revoked or expired claim information for different use cases.
V19. **Inadequate anonymization** [24]	It occurs when systems fail to adequately anonymize or separate instances of credential presentations, allowing different instances belonging to the same Holder to be linked together.
V20. **Insufficient session management** [43]	It refers to weaknesses or flaws in the way a Verifier handles user sessions, making it vulnerable to attacks such as session hijacking, session fixation, and other forms of unauthorized access. This can include several specific issues such as lack of secure connection or lack of session timeout.
V21. **Insecure data exchange protocol** [54]	It arises from the absence of essential security features, like the use of fresh nonces, within exchange protocols. Such omissions can pave the way for severe security breaches, including man-in-the-middle (MitM) attacks and credential replay.
V22. **Improper protocol parameters verification** [30]	It arises from the failure to adequately verify, validate, and sanitize the parameters used in communication protocols. For example, insufficient validation of redirect URIs in OAuth requests can allow attackers to redirect users to malicious sites under their control to steal user's data.
V29. **Excessive data exposure** [41,63]	This vulnerability occurs when an application API inadvertently exposes sensitive information, such as PII or authentication credentials. This vulnerability can be introduced due to various factors, including insecure coding practices, misconfigurations, and lack of input validation.
V32. **Lack of exact redirect URI matching** [20]	The complexity of implementing and managing pattern matching correctly obviously causes security issues. If the authorization server does not sufficiently validate the redirect URI, it can break client identification/authentication.
V33. **Lack of sufficient entropy for request_uri** [20,39]	It arises when the implementation does not provide the request_uri with enough entropy.
V34. **Lack of proper Wallet invocation** [17]	This vulnerability arises when the implementation lacks a mechanism to properly invoke the legit Wallet on the Holder's device

B Controls

In the following, we provide the definition of Controls that are presented within Table 1 and Table 2. Note that, the controls that do not have a reference are self-defined that are extracted based on the requirements that are defined in [13].

Table 4: Controls

C1. **Secure logging mechanism** [38]	Logs with timestamps are required to prove actions performed by entities. They must not include PII to avoid leaking sensitive information.

C4. **Secure storage** [42]	Information like private keys or credentials are stored securely to prevent unauthorized access. For example, a Trusted Execution Environment (TEE) is a secure area of the processor that could be used to store private keys and access them with increased security.
C11. **Holder revocation of Credentials**	Holders have a mechanism to remotely revoke Credentials, in case the Wallet becomes inaccessible (e.g. stolen, lost) or compromised.
C13. **Notification to inform the Holder of any Credential request/usage**	Notifications are used to inform Holders every time a Credential is requested on their behalf or a credential they own has been presented to a Verifier. Holders are also able to block malicious activities.
C18. **Secure communication** [46]	Communication between two entities is encrypted and authenticated (e.g., by using the latest version of TLS).
C23. **Validation of request/response messages** [50]	Entities validate the correctness and signatures of request/response messages to ensure their authenticity.
C26. **Privacy by design** [16]	Entities adopt privacy-by-design principles that treat privacy as a core consideration throughout the entire lifecycle of credentials, encompassing issuance, presentation, revocation, and deletion. This involves following best practices such as minimizing data disclosure, obtaining user consent, and leveraging techniques like Zero-Knowledge Proof.
C32. **Key binding** [32,33]	The Verifier checks the Holder's trustworthiness using key binding methods before accepting the credential presentation.
C33. **Exact redirect URI matching** [31]	The implementation is advised to simplify the required logic and configuration by using an exact redirect URI matching. This means the authorization server ensures that the two URIs are equal.
C34. **Short/one-time use request_uri** [30]	The implementation is using Request Object URI that is short-lived and preferably one-time use.
C35. **Request_uri value randomness** [30]	The Request Object URI is created by including a cryptographic random value of 128 bits or more.
C36. **Server-provided nonce** [32]	The implementation is creating the Wallet proof using a server-provided nonce.
C37. **Use of sender-constrained access token** [31,23]	The implementation uses a sender-constrained access token by using DPoP or mTLS to bind the access token to its sender.
C38. **Use of State/Nonce** [31]	The traditional defense against CSRF attacks is the OAuth "State" parameter. However, as the Nonce value is transaction-specific as well, it can provide a way to map the request and responses, thus mitigating the attack.
C39. **Authorization request validation** [23]	In case of including the authorization request parameters directly in the URL, the authorization request needs to include all the parameters defined in Section 5 of [32] and verify them as defined in RFC6749 [19]. In the case of a PAR request, the request needs to include all the parameters defined in PID/(Q)EAA credential issuance [23] and Verify them as defined in Steps 5 and 6 of [23].
C40. **Token request validation** [23]	The Issuer upon receiving the token request is needed to check that the request contains all the needed parameters and verify them as defined in Step 14 of [23].
C41. **Credential request validation** [23]	The Issuer upon receiving the credential request is needed to check that the request contains all the needed parameters and verify them as defined in Step 18 of [23].
C42. **Wallet trust evidence request validation** [23]	The Wallet Provider upon receiving the Wallet Trust Evidence request is needed to check that the request contains all the needed parameters and verify them as needed in Steps 13-17 of Wallet Trust Evidence request in [23].
C43. **Presentation request validation** [54]	The Wallet upon receiving a presentation request by the Verifier is needed to check (1) if the Verifier is trusted, and (2) check if the request contains all the necessary parameters as defined in Section 5 in [54] and verify them.
C44. **Presentation response validation** [54]	The Wallet upon receiving a presentation request by the Verifier is needed to check (1) if the Verifier is trusted, and (2) check if the request contains all the necessary parameters as defined in Section 5 in [54] and verify them.
C45. **Authorization response validation** [32,23]	The Wallet upon receiving the authorization response is needed to check that the response contains all the needed parameters and verify them as defined in Section 5.2 of [32] and Steps 10-11 of PID/(Q)EAA issuance of [23].
C46. **Token response validation** [32,23]	The Wallet upon receiving the token response needs to check the response based on Section 6.2 of [32] and Step 15 of PID/(Q)EAA issuance of [23].
C47. **Credential response validation** [32,23]	The Wallet upon receiving the credential response needs to check the response based on Section 7.3 of [32] and Steps 19-21 of PID/(Q)EAA issuance of [23].

C48. **Wallet trust evidence response validation** [23]	The Wallet upon receiving the Wallet Trust Evidence response is needed to perform the integrity check by verifying the signature, checking it contains the mandatory parameters, and verifying the values (e.g., issuer identifier is equal to the Wallet Provider that it starts the interaction with) based on Wallet Trust Evidence response in [23].
C49. **Using iss parameter** [31]	The Authorization Server implementation returns an issuer identifier in the authorization response that is verified by Wallet.
C50. **Using universal app links** [49,32]	The Wallet implementation is using Universal app links as a way for secure wallet invocation [32].
C51. **Use of Proof Key for Code Exchange** [49,31]	For OAuth flows, particularly in public clients, implement `PKCE` to mitigate the risk of authorization code interception and misuse.

C Summary Table of Threats

Table 5: Threats to verifiable credentials and verifiable presentations.

Threats	**Affected Entity**	**Affected Asset**	**Description**
T1. Theft of sensitive information [27,56,37,18,2,45,25,21,9,34]	Issuer, Verifier, Holder, Wallet Provider	IVHWP-DA	This threat involves the unauthorized access or exposure of personal, confidential, or critical information.
T2. Unavailability [27,7,37,18,2,45,25,21,9]	Issuer, Verifier, Wallet, Wallet Provider	IVWPW-SA	This threat targets the availability of services, making them inaccessible to legitimate Holders.
T3. Configuration modification [18]	Issuer, Verifier, Wallet, Wallet Provider	IVWPW-SDA	This threat category covers actions that undermine the trustworthiness of data and systems.
T4. Unauthorized access [37,34,32,31]	Issuer, Verifier, Wallet Provider, Wallet	IVWPW-DA	This threat involves malicious actors intending to gain unauthorized access to information or to compromise a system.
T5. Entity impersonation [27,37,45,8,34]	Issuer, Verifier, Wallet, Holder, Wallet Provider	IVHWPW-SDA	This threat involves malicious actors playing the role of a legitimate entity (e.g., Issuer).
T6. Repudiation of requested/obtained/revoked verifiable credentials/Wallet Trust Evidence [18,45]	Issuer, Verifier, Holder, Wallet Provider	IVHWP-SDA	This threat involves an entity denying the initiation or receipt of a transaction or an action.
T7. Credential theft [8]	Verifier, Holder	VH-DA	This threat refers to the malicious acquisition of the Holder's credentials to present it to the Verifier and obtain unauthorized access to the service.
T8. Metadata tampering [32]	Issuer, Verifier	IV-SDA	This threat targets entity Metadata, intending to configure all or some of the endpoints to point to the ones under the attacker's control.
T9. Issuer/Verifier operator deception [7]	Issuer, Verifier, Holder	IVH-IA	This threat involves malicious actors trying to deceive the Issuer/Verifier operator in an in-person credential issuance/presentation scenario that the identity verification/credential check has been successful.

T10. Issuer abuse [18]	Issuer, Holder	IH-DSIA	Issuer Abuse is a multifaceted threat that arises when individuals with access to critical systems within an organization misuse their positions of trust and authority. An example of this abuse can be illegitimate credential revocation or the usage of a user's identity information for performing unauthorized actions for personal gain.
T11. Tampering of a valid credential [7,18,34]	Verifier	V-SA	This threat involves the unauthorized alteration or modification of credential data.
T12. Use of a stolen credential [15]	Verifier	V-SA	This threat involves the act of using leaked or stolen credentials.
T13. Use of an expired/revoked/ephemeral credential [7]	Verifier	V-SA	This threat involves the act of using credentials that are no longer valid or have been revoked.
T14. Use of inaccurate or outdated claims in credential [34,15]	Verifier	V-SA	This threat involves the act of using credentials that have incorrect, misleading, or outdated claims.
T15. Exploitation of faulty credential status check mechanism [15]	Holder	H-DA	This threat occurs when the credential validity check mechanism is not designed in a privacy-preserving way (e.g., it demands direct communication between the Verifier and Issuer). Thus, it would leak sensitive information about the usage of credentials to the Issuer and enable Holder's tracking.
T16. Unauthorized presentation response (vp_token) forwarding [15]	Verifier	V-SA, WV-IA	This threat aims to impersonate the user at the Verifier by forwarding the VP token extracted from the interaction of Wallet and Verifier into the attacker session with a verifier under his own control.
T17. Unauthorized presentation request forwarding [32]	Verifier	V-SA	This threat aims to deceive the victim into completing the presentation phase with a Verifier on behalf of an attacker.
T18. Registration of non-compliant entity	Issuer, Verifier, Wallet Provider	IVWP-DSIA	This threat aims to register a non-compliant entity to operate in the ecosystem for a malicious purpose.
T19. Exploitation of faulty data exchange protocol implementation	Holder	H-DA	This threat considers improper implementations without following the best practices that can lead to increasing the attack surface.
T20. Trusted list compromise [7,18]	Issuer, Verifier, Wallet Provider	IVWP-DA	This threat aims to manipulate the list of trusted entities in the ecosystem, to change an untrusted entity to the trusted instance.
T21. Exploitation of faulty key generation mechanism [56,21]	Wallet, Holder, Issuer, Verifier	IVHW-DSIA	This threat pertains to vulnerabilities in the process of generating cryptographic private keys, where the randomness and secrecy of the seed value are compromised.
T22. Wallet compromise [27,56,37,18,45,25,21,9,8]	Wallet	W-DA	This threat underscores the risk of unauthorized access and manipulation of a device's data and functionality.
T23. Device loss [9]	Wallet	H-DA	This threat scenario involves the physical loss or theft of a user's device, such as a smartphone, which gives an attacker unhindered access to the device, and if there's an insecure wallet unlock mechanism, access to the wallet data on the device as well.
T24. Malicious app [28]	Wallet	W-DSA	This threat scenario involves the threat actor deceiving the Holder into installing a malicious app with the aim of unauthorized access to the Holder's wallet.
T25. Backup file disclosure [28]	Wallet, Holder	WH-DA	This threat enables the threat actor to misuse the insecure Wallet backup functionality with the aim of unauthorized access to the Holder's backup.

T26. Cryptographic key/credential leakage [27,28]	Wallet, Holder	WH-DA	Insecure credential storage refers to the improper handling and storage of the credentials/private keys within the Wallet.
T27. Unauthorized access via debug interface [28,21]	Wallet	W-DA	This is a security threat where attackers exploit debugging interfaces or modes of a system, application, or device to gain unauthorized access or control.
T28. Exploit weak key recovery mechanism [27,18]	Wallet	W-DSA	This threat refers to the malicious act of targeting systems or procedures designed for the recovery of cryptographic keys, intending to steal private keys.
T29. Illegitimate credential revocation [18]	Wallet	W-SA	This threat refers to an unsecured identity revocation mechanism that could be misused by an attacker.
T30. Data minimization violation [18,25,34]	Holder	H-DA	Excessive personal data collection refers to the practice of requesting, obtaining, and storing more personal information from individuals than is strictly necessary for the intended service or transaction.
T31. Exploit faulty user consent mechanism [15]	Wallet, Holder	H-DA, W-SA	This threat is related to the improper design of the User's consent mechanism that can be misused by the threat actor for unauthorized access to User's credential, or share User's credential without user awareness.
T32. Exploit faulty credential refresh/update mechanism [15]	Holder, Issuer	I-SA, H-DA	This threat is related to the faulty design of the credential refresh mechanism that may enable Holder activity profiling.
T33. Illegitimate credential/sensitive data storage [15]	Holder	H-DA	This threat involves malicious storage of credential presentation by the Verifiers.
T34. Exploit faulty logging mechanism	Issuer, Verifier, Wallet Provider, Wallet	IVWPW-SA	This threat occurs if the logging mechanism is insecure and not based on best practices.
T35. SIM swap fraud [45]	Wallet, Holder	W-SA, H-DA	This threat targets the Holder's mobile phone carrier with the aim of transferring the victim's phone number to a SIM card controlled by the threat actor.
T36. Linkability [58]	Issuer, Holder	IH-DA	This threat targets the privacy of Issuers and Holders intending to link different instances of credential presentations belonging to the same Holder.