Baltic J. Modern Computing, Vol. 8 (2020), No. 4, pp. 638–674
https://doi.org/10.22364/bjmc.2020.8.4.11

Corda Security Ontology: Example of Post-Trade Matching and Confirmation

Mubashar IQBAL, Raimundas MATULEVIČIUS

Institute of Computer Science, University of Tartu, Tartu, Estonia
{mubashar.iqbal, rma}@ut.ee

Abstract. Blockchain technology is ready to revolutionise the financial industry. The financial industry has various security challenges *(e.g., tampering, repudiation, denial of service, etc)*. Also, the domain of information security has problems related to conceptual ambiguity and the semantic gap. The Corda platform provides suitable technological infrastructure to build the blockchain-based application (CorDapp) in the financial industry to overcome security challenges. In this paper, we build a Corda-based security ontology *(CordaSecOnt)* to improve the security of financial industry from an ontological analysis that combines blockchain-based Corda platform. We use Web ontology language (OWL) to build a semantic knowledge base to eliminate conceptual ambiguity and semantic gap in information security. Our ontology provides classifications of assets, security criteria, threats, vulnerabilities, risk treatments, security requirements, countermeasures and their relations. We evaluate the ontology by performing security risk management (SRM) of capital market post-trade matching and confirmation.

Keywords: Corda security ontology, CorDapp security risk management, Corda security risks analysis, CordaSecOnt, Blockchain-based application

1 Introduction

The advent of *Blockchain* technology introduces new concepts to revolutionise the financial industry. The European Central Bank (2017) presents the financial industry interests to use blockchain technology in their infrastructure. The centralised infrastructure of financial industry is challenged by strict regulations and constant security risks (AL-essa, 2019). The security risks harm the business processes and valuable assets that could lead to a reputation loss or financial sanctions for the organization. For example, the hackers used SWIFT credentials to steal $81m from the Bangladesh bank (Polyviou et al., 2019). Also, the financial industry ranks second in data breaches. Hence, the security risks in capital markets enable manipulative, illegal and abusive trade practices (WEB, a). This paper focuses on improving the security of the financial industry

from an ontological analysis by combining blockchain and blockchain-based Corda platform. The ontology includes the case of post-trade matching and confirmation, and Corda-based application (CorDapp) to overcome various security challenges of it.

Other problems are conceptual ambiguity and the semantic gap in the field of information security. Fenz et al., (2009) and Mozzaquatro et al., (2016) explain that information security is one of the top challenges and required common understanding to implement SRM approaches. The ontology describes the concepts of a particular domain that builds on a controlled vocabulary. In this study, we use Web ontology language (OWL) to build a semantic knowledge base to eliminate conceptual ambiguity and a semantic gap in information security when building financial industry CorDapps. The CordaSecOnt provides unified and formal knowledge models that could support the SRM, clear understanding and communication of CorDapp information security.

The field of blockchain is continuously evolving, and security plays an important role in the acceptance of blockchain-based applications. In the security domain of blockchain, there are various interchangeable security concepts that bring the conceptual ambiguity and confusion to treat security threats effectively. An ontological representation of information security is a valuable tool to assess and communicate the security aspects of the application that brings timely decisions to fix them.

The research objective of this paper is to build Corda-based security ontology *(CordaSecOnt)* for capital markets post-trade matching and confirmation. It provides an extensible knowledge base of information security including the notions of Blockchain and Corda. We establish a road-map (Fig. 1) to reach our objective. First, we define the research objective. Second, we align the concepts of Blockchain, Corda and information security. Third, we follow the domain model of SRM to build CordaSecOnt.

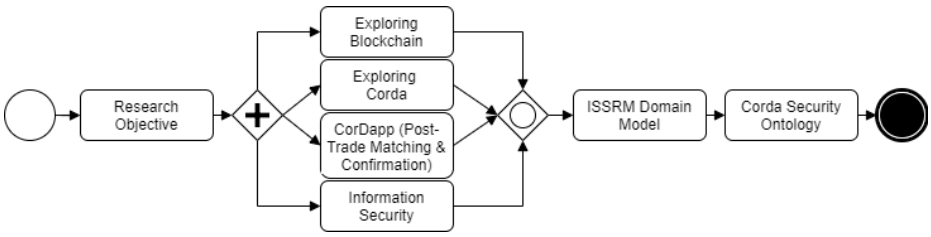

Fig. 1. The road-map to build Corda-based security ontology

This paper is an extension of the work reported in (Iqbal and Matulevičius, 2020), where we perform SRM of post-trade matching and confirmation. In this paper, we present the Corda-based security ontology and its application in post-trade matching and confirmation case. The paper is structured as follows: Section 2 presents the Background and Section 3 discusses the capital markets post-trade matching and confirmation in the context of centralised and decentralised infrastructure. Section 4 presents the ontology construction and in Section 5 we perform an evaluation of ontology. Section 6 is the related work and Section 7 concludes the paper.

2 Background

In this section, firstly, we discuss the Blockchain, Corda platform and CorDapp. Secondly, we present the concepts of security risks analysis and ontology building tools.

2.1 Blockchain

Blockchain is an append-only decentralised distributed ledger technology that promises to overcome the security challenges and enhance data integrity. Blockchain operates over a peer-to-peer (P2P) network where the nodes join the network and establish a chain of blocks. In Blockchain, a block is connected to a previous block by a unique cryptographic hash (Fig. 2). The ledger is immutable and updates every time over a P2P network when a new block populates.

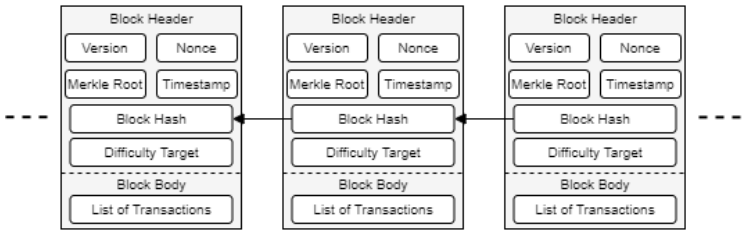

Fig. 2. The Block structure of Blockchain

Fig. 2 illustrates a block structure that contains a block header and a block body. Block header has a version number, timestamp, block size, difficulty, nonce, and the number of transactions. Block body contains confirmed and processed transactions in a Merkle tree. Blockchain is classified as a permissionless or permissioned (Pradeepkumar et al., 2018). In a permissionless blockchain, anyone can join the network and participate in the consensus mechanism. Also, the transactions are publicly visible to every participant node. In permissioned blockchain, only predefined verified nodes can join the network and participate in the consensus mechanism. The transaction visibility is restricted (Ali et al., 2018) in permissioned Blockchain.

In a Blockchain, a smart contract (SC) is a computer program (Atzei et al., 2017), (Buterin, 2014) which constitutes a digital contract to store data and execute (Macrinici et al., 2018) when certain conditions meet. For example, in the Ethereum platform, developers use *Solidity* programming language to write SC and to build decentralized applications (dApps) (Buterin, 2014). In Hyperledger Fabric (HLF), SC is known as Chaincode. Similarly, other blockchain platforms introduce SCs to perform contractual agreements in a digital realm (Iqbal and Matulevičius, 2019a). The SCs are the high-level programming language-based programs and those can be error-prone where security flaws could be introduced (e.g. the reentrancy bug (Liu et al., 2018)).

Blockchain eliminates the trusted intermediary and follows the decentralized consensus mechanism to validate the transactional information. For example, Bitcoin and Ethereum use Proof of Work (PoW) consensus, that is a widely used computational

rich energy-waste consensus strategy where special nodes called miners define the state of the ledger by solving the crypto puzzle. In contrast, Proof of Stake (PoS) is an energy-efficient consensus strategy (Zheng et al., 2016) where miners become validators (WEB, b). In order to participate in the consensus, validators lock a certain amount of cryptocurrency. There are other consensus mechanisms, for example, Delegated Proof of Stake (DPoS), Proof of Authority (PoA), Proof of Reputation (PoR) and Proof of Spacetime (PoSt). Table 1 shows the comparison of Blockchain platforms.

Table 1. Comparison of Blockchain platforms based on different characteristics

	Bitcoin	**Ethereum**	**HLF**	**Corda**
Type	Permissionless	Permissionless	Permissioned	Permissioned
Smart contract	Yes	Yes	Yes	Yes
SC language	Script	Solidity	GO, Java	DAML
Consensus	PoW	PoW, PoS	PBFT, CFT	Validity, Uniqueness
Cryptocurrency	Bitcoin (BTC)	Ether (ETH)	No cryptocurrency	No cryptocurrency
Transactions	7 TPS	8-9 TPS	Thousands	Thousands
Confidentiality	No	No	Yes	Yes
Applications	Cryptocurrency	dApps	dApps	Financial dApps

2.2 Corda platform and CorDapp

Corda is an open-source enterprise blockchain-based platform and CorDapp is a Corda-based decentralised application. Corda is a permissioned blockchain and nodes require certain permissions to access the data. Corda focuses on to bring privacy, transparency and security to financial operations between different parties. The Corda platform uses i) validity, and ii) uniqueness consensus (Koens et al., 2019), (Hearn, 2016). The validity consensus ensures the correctness of input & output states and required signatures in a transaction. The uniqueness consensus checks if the transaction is not already consumed (e.g., protection against double-spending) (Koens et al., 2019), (Hearn, 2016).

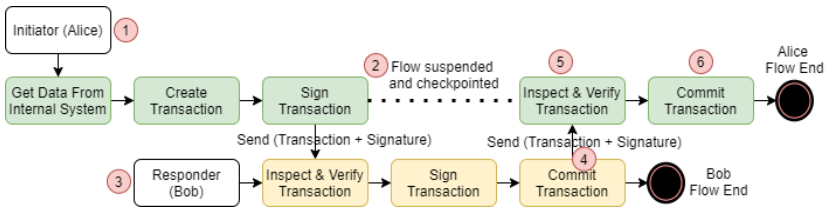

Fig. 3. Transaction flow of Corda [adapted from: (Parag, 2020)]

In Corda, peers communicate point-to-point (Fig. 3). For example, the transaction specifies a list of message recipients that sign and verify the transaction. The flow mechanism supports multi-step coordination and verification of transaction messages. Corda network peers communicate back-and-forth before a transaction commit. Transaction flows validate the sequence of steps that makes a transaction legitimate and commit on the ledger that replicates over different peers on the P2P network. In Fig. 3, Alice

initiates the transaction flow and creates a transaction. Alice signs the newly created transaction and sends it to Bob. In this stage, Alice flow is suspended & checkpointed, and now Bob transaction flow begins. Firstly, Bob inspects and verifies the transaction upon receiving from Alice. Secondly, Bob signs a transaction and commits. The transaction now signed by both parties (Alice and Bob) and the system sends it back to Alice for verifying the signature of Bob. After verification of Bob signature, Alice commits the transaction that fulfils the criteria of a successful transaction process and completes the flow. This is an example of two parties transaction process, but Corda flows can involve multiple parties and multiple signatures (Parag, 2020).

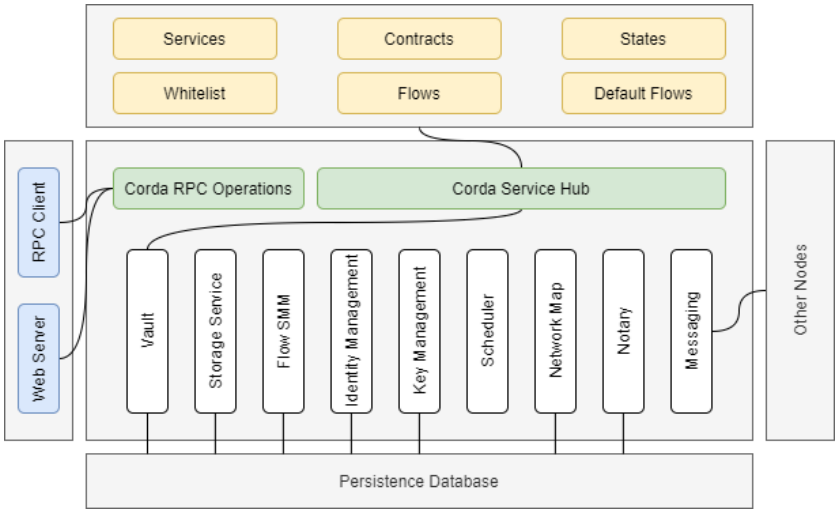

Fig. 4. Architecture of Corda platform [(adapted from: WEB c)]

The architecture (Fig. 4) and Table 2 describes the core elements of Corda platform. A Corda node is a JVM run-time environment, it has a unique identity on the network and hosts Corda services and CorDapps. CorDapp is installed at the level of the individual node, rather than on the network itself (WEB, d). Corda uses an asynchronous protocol (e.g., AMQP/TLS) (WEB, d) when nodes communicate with each other and HTTP communication for registering a Corda node. In Corda, client application use RPC calls to communicates with Corda nodes and Corda vault is a database that relies on java database connectivity from the Corda node (WEB, c), (WEB, d).

2.3 Security risks analysis

Firstly, we explore *SRM Domain Model* (Fig. 5) that supports making decisions related to information security of the system (Dubois et al., 2010), (Matulevičius, 2017). SRM domain model comprises three main concepts: (*i*) asset-related concepts, (*ii*) risk-related concepts, and (*iii*) risk treatment-related concepts.

Table 2. Components of Corda platform

Component	Description
Persistence layer	A persistence layer for storing data, for example, in SQL-based database.
Network interface	A network interface for interacting with other nodes.
RPC interface	An RPC interface for interacting with the node's owner.
Service hub	A service hub for allowing the node's flows to call upon the node's other services that are node utilities.
CorDapp interface	CorDapp interface and provider for extending the node by installing CorDapp.
Vault	Stores output states relevant to a particular node.
Transaction storage	Key value store for attachments, transactions, and serialized state machines.
Flow State Machine Manager	Manages operation of flow state machines. Flows are routines to run and update the ledger for the node. States are the facts to reach agreement.
Contracts	Contracts defining what constitutes a valid ledger update.
Identity/Key Management	Manages various supported identities and generated keys used to sign transactions.
Scheduler	Schedules operations for future points in time.
Network Map	Searchable phone book of nodes on network.
Notary	Obtains authorized signatures.
Messaging	Interface with other nodes.

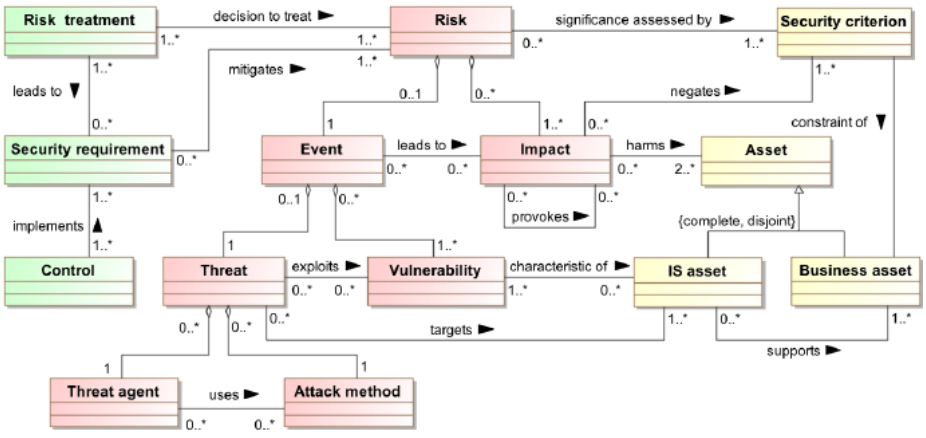

Fig. 5. The SRM domain model [adapted from: (Dubois et al., 2010), (Matulevičius, 2017)]

The assets could be classified as system asset or business asset. The business asset has value and system asset supports it. Security criteria (C - Confidentiality, I - Integrity and A - Availability) distinguish the security needs. In risk-related concepts, the risk is a combination of risk event and impact. The risk event constitutes the threat and one or more vulnerabilities. The threat targets the system asset and it is triggered by the threat agent, who uses an attack method and exploits the vulnerability. Impact harms the asset and negates the security criteria. The risk treatment-related concepts present decisions to treat security risk by defining security requirements. Security requirements are implemented as the controls (e.g., countermeasures). We extract the object properties (e.g., relations) (Table 3) from the SRM domain model. These relations are associated with assets, risks and risk treatment-related concepts. In risk-related concepts, we only utilise the concepts of *threat* and *vulnerability* that refine the scope of our ontology to enhance the understandability of CordaSecOnt to a wide range of domain experts.

Table 3. Ontology relations from the SRM domain model

Relation	**Description**
characteristicOf	A vulnerability is a characteristic of a one or more system assets that exposes weakness. For example, a vulnerability "MissingAccessControl" is a characteristic of system asset "ImportMessageEngine".
constraintOf (hasConstraint)	Security criteria is a constraint of business assets. In CordaSecOnt, constraintOf relation is an inverse of hasConstraint. For example, a business asset "ImportMessage" has a constraint "Integrity".
exploits	A threat exploits zero to several vulnerabilities. For example, a threat "Spoofing" exploits "MissingAccessControl" vulnerability.
implements	One or more countermeasures implements one or more security requirements. For example, a countermeasure "AccessControl" implements "RestrictUnauthorisedAccess" security requirement.
leadsTo	Each risk treatment decision leads to the refinement of none or several security requirements. For example, a risk treatment decision "Reduction" leads to "RestrictUnauthorisedAccess" security requirement.
mitigates	A security requirement mitigates one or more security threats. For example, a security requirement "RestrictUnauthorisedAccess" mitigates a threat of "Spoofing".
negates	At the level of business assets, vulnerability negates the defined security criteria of business assets. For example, a vulnerability "MissingAccessControl" negates (Availability or Confidentiality or Integrity) of business assets.
supports	A system asset supports one or more business assets. For example, a system asset "ImportMessageEngine" supports business assets "ImportMessage and TradeMatching".
targets	A threat targets one or more system assets. For example, a threat "spoofing" targets "ImportMessageEngine and SenderEngine" system assets.

The SRM domain model explains relationships among assets, risks and risks treatments related concepts that establish a process (Fig. 6) combining different activities to perform SRM. For example, in this work, we explore the case of post-trade matching and confirmation to identify what assets to secure. Then, we determine the security objective in the context of confidentiality, integrity and availability. Following the process,

we perform a security risk analysis to identify what are the threats within post-trade matching and confirmation that exploit various vulnerabilities to harm the assets and negate security criteria of the business assets. In the next stage, we define the risk treatment decisions to treat the threats and determine the security requirements. Finally, the security requirements are implemented by incorporating security controls. SRM is not a static approach, hence the process model of SRM enables to perform several iterations to reach an acceptable level of each risk (Matulevičius, 2017).

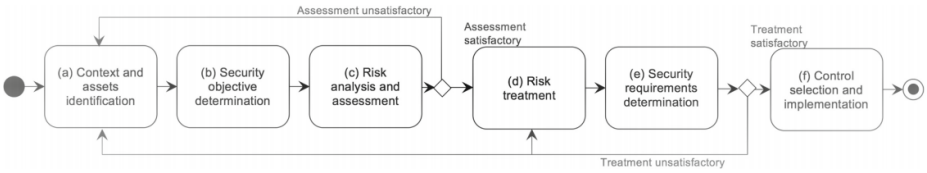

Fig. 6. The process model of SRM domain model [adapted from: (Matulevičius, 2017)]

Furthermore, we utilise the STRIDE (Ruffy et al., 2016) *(S - Spoofing, T - Tampering, R - Repudiation, Id - Information disclosure, D - Denial of service, E - Elevation of privileges)* threat model (Table 4) that supports a systematic analysis to identify and explain the potential security threats. Here, STRIDE is selected as a security threat analysis tool to analyse the security of centralised post-trade matching and confirmation. STRIDE supports the security risk analysis activity of the SRM process model.

Table 4. STRIDE threat model

Threat	**Description**
Spoofing	Pretending to be someone or claiming a false identity.
Tampering	Unauthorised modification in data, process, memory or network.
Repudiation	Denying that a specific action is not performed by you or you are not responsible for that action. For example, deny if performed a destructive action.
Information disclosure	Data leaks or disclosing information to an unauthorised user.
Denial of service (DoS)	Exhausting system resources to make service or a network unavailable for the potential system users.
Elevation of privilege	Gaining unauthorised access and allowing one to perform such operations that are not authorised to do.

2.4 Ontology building tools

Ontology deals with the nature of existence. According to the Oxford dictionary, ontology combines a list of "concepts and categories in a subject area that shows the relationships" between them (WEB, e). An ontology brings generalisation in a specific domain and establishes an exchange of information (Noy and McGuinness, 2001). Ontology structure area of interest (Mozzaquatro et al., 2015) and elaborates the meaning

of concepts along with their relations that support to overcome the consequences of a misunderstanding that could be time-consuming and costly. In (Noy and McGuinness, 2001), the authors listed some reasons to explain why to develop an ontology? For example, the Table 5 explains the same reasons in the perspective to develop CordaSecOnt.

Table 5. Ontology building concepts

General reasons to build ontology	**CordaSecOnt perspective**
To share common understanding of the structure of information among people or software agents.	To share common understanding of the information security of CorDapp among security experts.
To enable reuse of domain knowledge.	To enable reuse of CorDapp information security domain knowledge.
To make domain assumptions explicit.	To make explicit specifications of CorDapp information security domain knowledge for new users.
To separate domain knowledge from the operational knowledge.	To describe a CorDapp information security knowledge from its ontology components according to a required specification.
To analyse domain knowledge.	Formalising CorDapp information security into an ontological knowledge domain that permits security experts to analyse security of CorDapp.

We use OWL to build Corda security ontology. OWL is based on description logic (DL) and a W3C (WWW Consortium) standard to build an ontology. OWL is a semantic web language to illustrate rich and complex knowledge about things, groups of things, and relations between things (OWL Working Group, 2012). OWL combines concepts (e.g., classes/subclasses) within a domain (e.g., Corda security domain), object properties (e.g., relationships of concepts), data properties, restrictions, individuals (e.g., instances of a class) and inference that is an automatic procedure to compute conclusions based on evidence and reasoning within an ontology.

DL deals with formal knowledge representation and provides a logical formalism for an ontology. DL illustrates the fundamental modelling concept relating to roles and concepts (Rector et al., 2004). DL-based knowledge includes two components: i) Terminological component (TBox), and ii) Assertion component (ABox) (Gao et al., 2013). The TBox is a conceptualisation that associated with a set of facts (e.g., ABox) within a knowledge domain. In Protégé, DL query tab is a plug-in feature to search knowledge from an ontology when it is consistent and reasoner is working.

OWL supports resource descriptive framework (RDF) to define metadata model (Hector and Boris, 2020), for example, RDF is a language that allows encoding, exchange and reuse of structured metadata (Miller, 1998) to build a readable semantic infrastructure (Hector and Boris, 2020). RDF supports triplet format (e.g., subject-predicate-object) for describing the ontology concepts. In this triplet *(Tampering **exploits** MissingAccessControl)*, *Tampering* threat is a **subject**, *exploits* is a relation that represent a **predicate** and *MissingAccessControl* vulnerability is an **object**.

We use SPARQL (SPARQL Protocol and RDF Query Language) as a semantic query language to retrieve and manipulate domain knowledge that is mapped in RDF

format (Herzog et al., 2007). SPARQL is W3C recommended query language to write semantic queries and use RDF middle-ware to get results from an ontology. For example, the following SPARQL query gets the system assets from the CordaSecOnt.

```
SELECT DISTINCT ?System_Asset  WHERE {
    ?System_Asset rdfs:subClassOf CordaSecOnt:SystemAsset
}
```

We build our ontology using Protégé ontology editor (Fig.7). Protégé is a free, open-source and most adapted ontology editor (Zamfira et al., 2018) that proposed by the Stanford University in California.

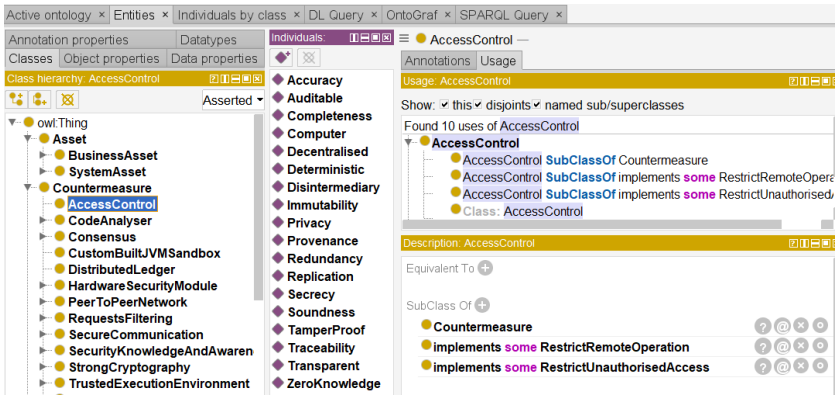

Fig. 7. Protégé ontology editor

3 Capital market post-trade matching and confirmation

In this section, we present the context of capital market post-trade matching and confirmation in centralised and decentralised infrastructure.

3.1 Context of centralised infrastructure

A capital market is a part of financial industry and it processes the financial instruments (e.g., securities, futures, options, and other assets). Mainly, the trading system has three modules: front-office, middle-office, and back-office module. The important action in the middle-office is *trade matching and confirmation*. The business process model (Fig. 8) represents the post-trade matching and confirmation process between counter-parties (*e.g., Bank X is a buyer and Bank Y is a seller*) (Iqbal and Matulevičius, 2020). The trade matching executes after receiving the trade details from front-office. The financial organization (*e.g., Bank X*) performs trade matching with *Bank Y*. The confirmation happens when the trade details are accepted and agreed by each counter-party. In centralised trading, trade matching is performed manually, so there are high

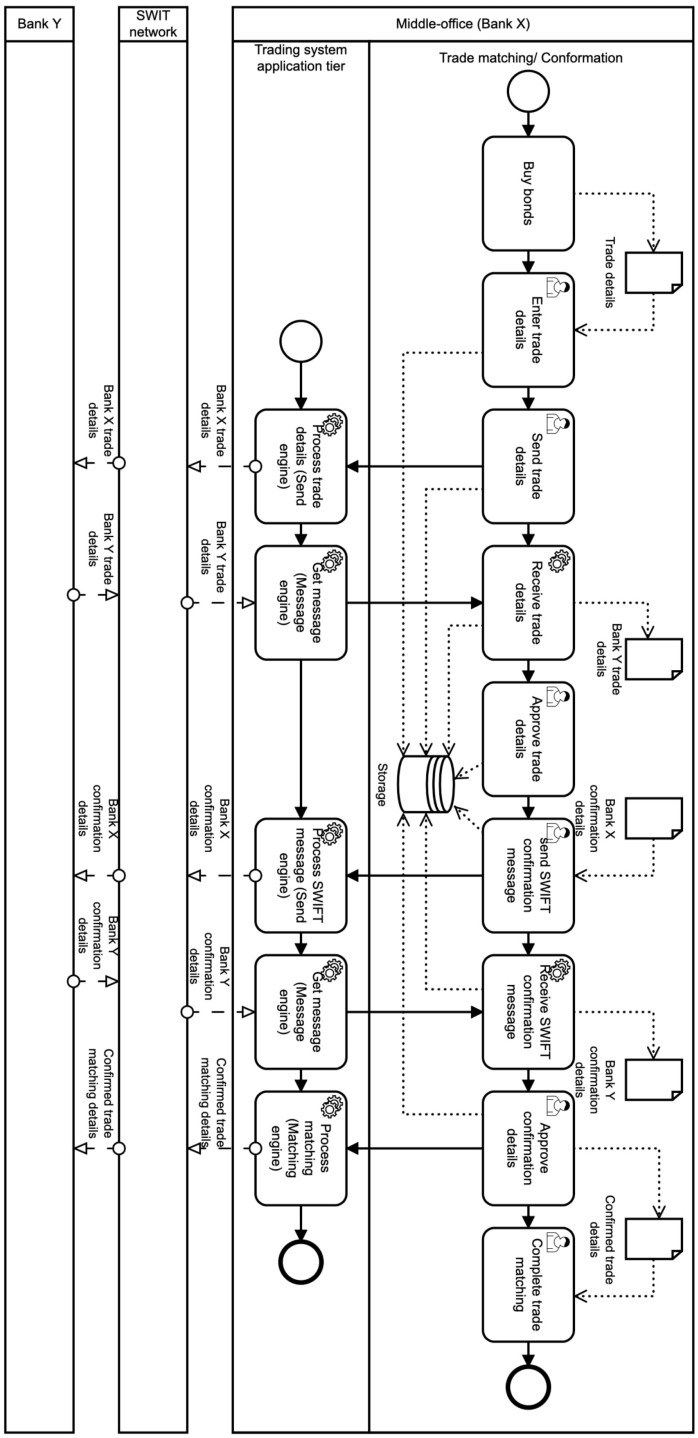

Fig. 8. Post-trade matching and confirmation in centralised infrastructure [adapted from: (Iqbal and Matulevičius, 2020), (Placāns, 2019)]

chances of security risks. The third parties (e.g., regulators and clearinghouses) assure that the trade is valid and all the necessary information is provided (Beker, 2015).

Similar to Iqbal and Matulevičius (2019b), the architecture of post-trade matching and confirmation in centralised infrastructure (Fig. 9) presents an abstraction of the system components that organised mainly in three layers. The *Presentation Layer* exposes the interaction interface where users interact with the application. The user interacts with the services and performs different operations, which are present in a *Application Layer*. The *Data Storage Layer* combines mainly the database, but also includes access rights details, trade details and logs.

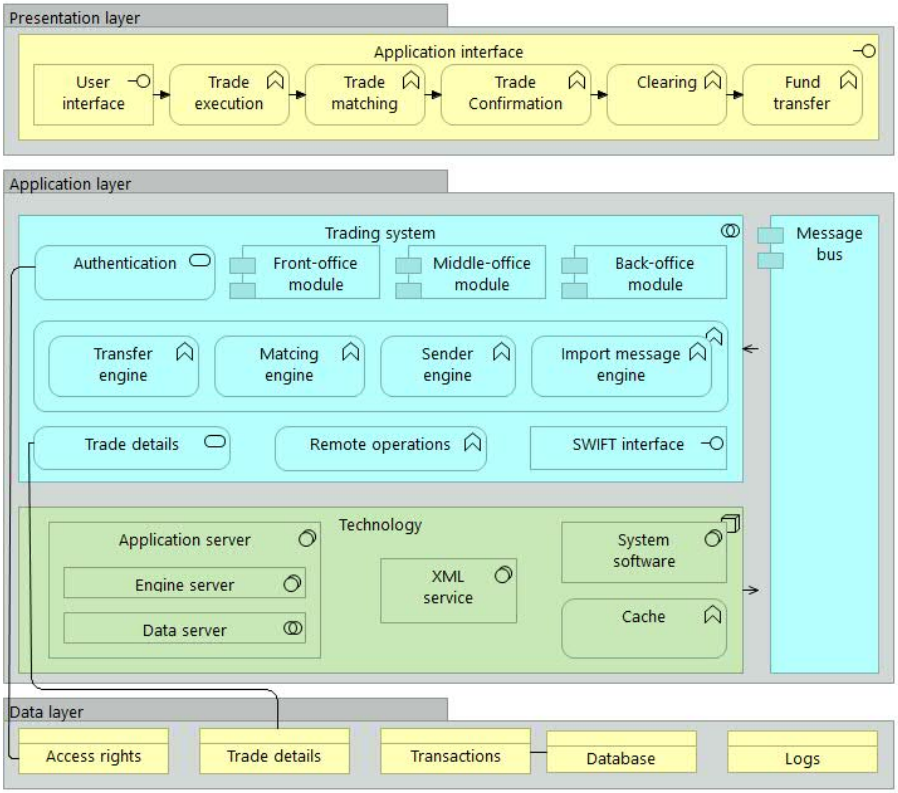

Fig. 9. Components of post-trade matching and confirmation in centralised infrastructure

3.2 Context of decentralised infrastructure

The business process model of CorDapp (Fig. 10) enables blockchain-based states, contracts, flows, and a vault to interact with counterparties to perform the assets exchange (Iqbal and Matulevičius, 2020). The counter-parties *(e.g., Bank X & Y)* perform the operations without relying on the manual operations and trusted third-party. The CorDapp performs the validation of a transaction by a notaries-based consensus (Hearn, 2016). The counter-parties receive validated data from distributed Corda ledger over a P2P

network. The involved counter-parties provide necessary details of the transaction by Corda node to CorDapp that validates and completes the post-trade matching and confirmation. The process increases efficiency and reduces the risks of manual operations. The architecture (Fig. 9) is extended to (Fig. 11) by integrating the Corda platform. Similarly, the architecture (Fig. 11) has three different layers along with the auxiliary components (e.g., client, nodes network, etc).

4 Ontology construction

We adopt the ontology construction method (Fig. 12) proposed by Uschold and Gruninger (1996), the method has five distinctive stages: i) Identify purpose and scope, ii) Building ontology, it includes capture, coding and integrating phases, iii) evaluation, iv) documentation and v) guidelines.

4.1 Ontology scope

We begin the ontology development by defining its purpose and scope. The purpose of an ontology is already discussed in *(Section 1)*. We follow the instructions of Noy and McGuinness (2001) to define the scope of ontology by answering these questions:

Q#1: What is the domain that the ontology will cover? The first question helps to determine the domain of our ontology. For example, the domain is ”information security of post-trade matching and confirmation using CorDapp”. We utilise the SRM domain model to elicit the relationships of security concepts (Table 3).

Q#2: For what we are going to use the ontology? The CordaSecOnt would help to perform SRM of CorDapp in two different perspectives. Firstly, what security threats are mitigated by using the CorDapp in post-trade matching and confirmation. Secondly, what security threats appear after using a Blockchain-based Corda platform.

Q#3: What types of questions the ontology should provide answers? The ontology during SRM of CorDapp would bring the answers to the following questions.

- What assets to secure?
- What are the security threats that are mitigated by CorDapp?
- What are the security threats that are appeared within CorDapp?
- What are the vulnerabilities that exploit by security threats?
- What security controls are in place to mitigate security threats?

Q#4: Who will use and maintain the ontology? Security professionals (e.g., domain experts) will use and maintain the ontology when building CorDapps in post-trade matching and confirmation.

These questions serve as the starting point to identify the scope of our ontology. To further refine the scope of our ontology, we utilise the relations from the SRM domain model *(discussed in Q#1)*. In Table 3, we provide the relations of SRM domain model that incorporated in our ontology. The SRM domain model relations are associated with asset, risks and risks treatments-related concepts.

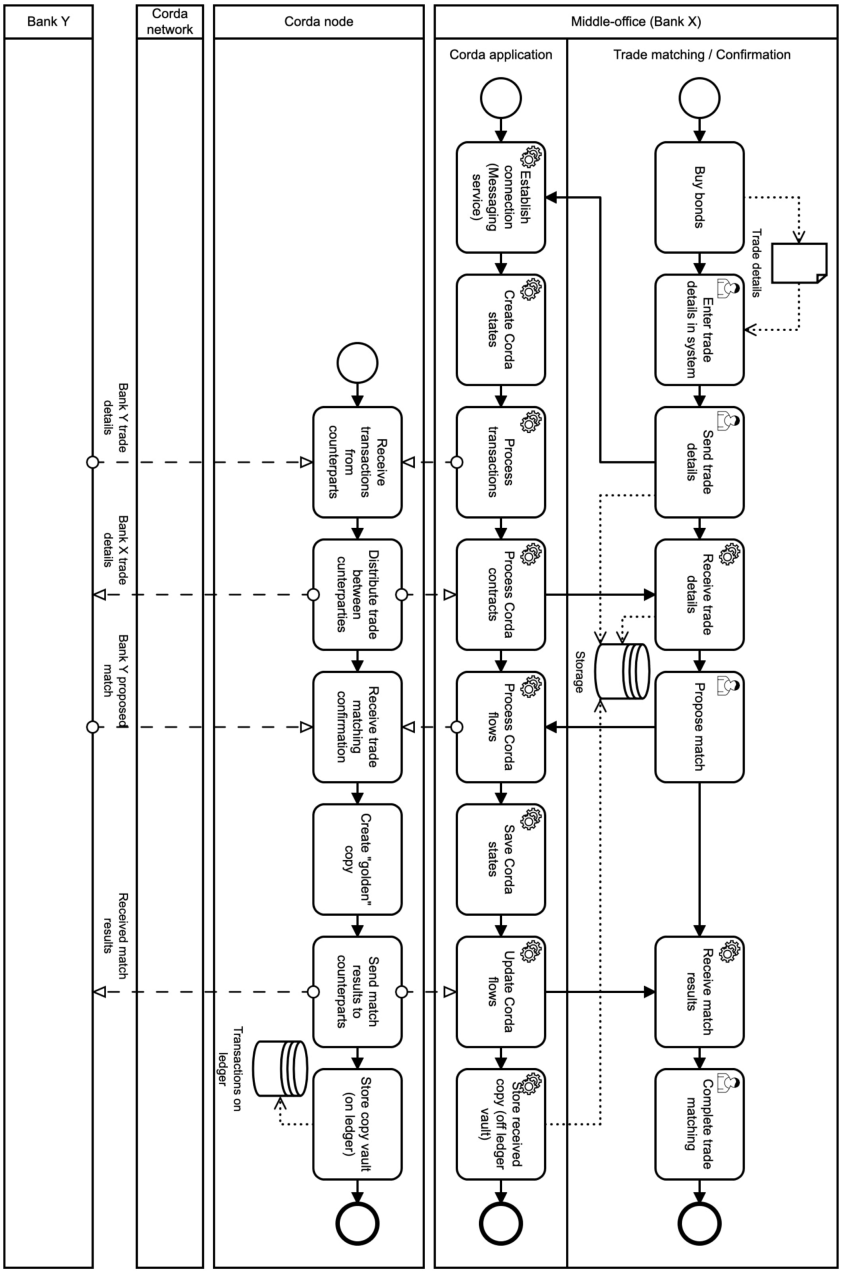

Fig. 10. Post-trade matching and confirmation in CorDapp [adapted from: (Iqbal and Matulevičius, 2020), (Placāns, 2019)]

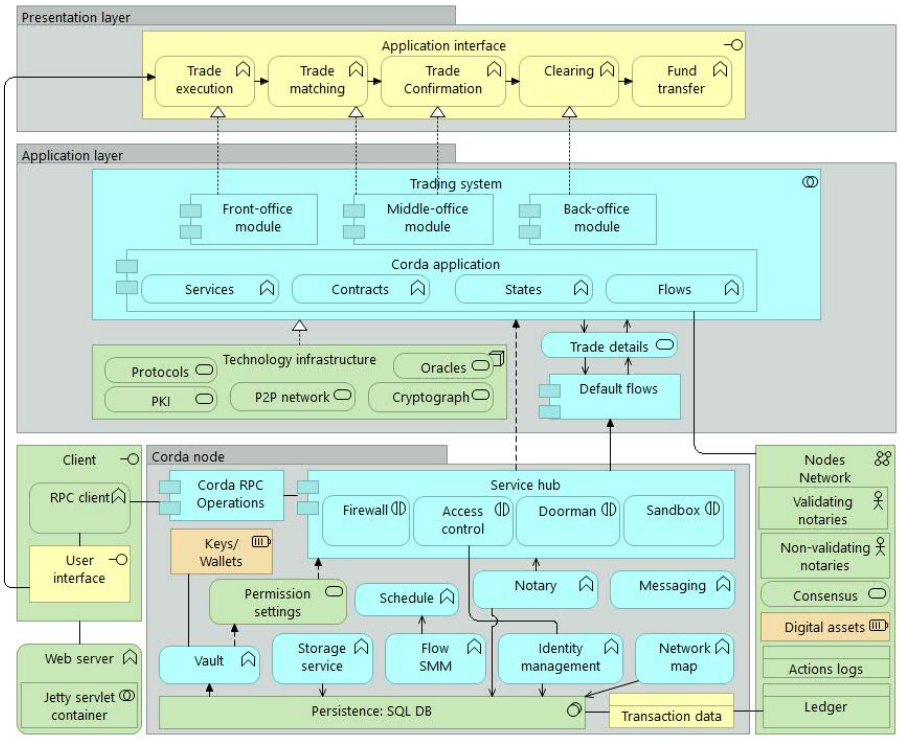

Fig. 11. Components of post-trade matching and confirmation in decentralised infrastructure

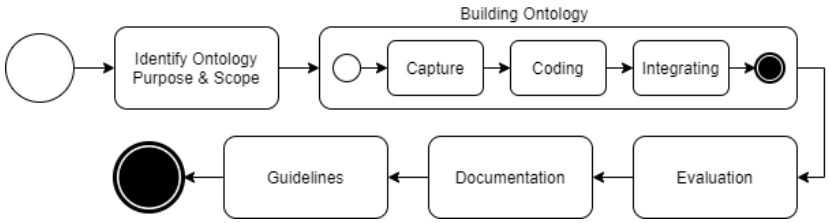

Fig. 12. Ontology construction method [adapted from: (Uschold and Gruninger, 1996)]

4.2 Ontology building

Firstly, we capture the domain information (e.g., concepts and relations) and classify in taxonomic structures. The classification technique refines the concepts belonging to assets, security criteria, threats, vulnerabilities, risk treatments, security requirements and countermeasures. The classifications improve the technical domain vocabulary of concepts. Secondly, we code the concepts and relations to formalise the domain knowledge in our ontology. The classifications related to *Assets*, *Threats*, *Vulnerabilities* and

Countermeasures are available in the following sections and classifications of *Security criteria*, *Risk treatments* and *Security requirements* are in Appendix.

4.2.1 Assets: We identify what assets to secure from the security threats, for example, assets have a value and need protection against security threats. The assets are classified as business (Fig. 13) or system assets (Fig. 14). Security criteria is a constraint of business assets and system assets support business assets. For example, a business asset "ImportMessage" hasConstraint Integrity. System assets "SenderEngine and ImportMessageEngine" support business asset "ImportMessage". The following statements are an example of DL for relations "hasConstraint" and "supports".

```
supports some BusinessAsset / hasConstraint some SecurityCriteria
```

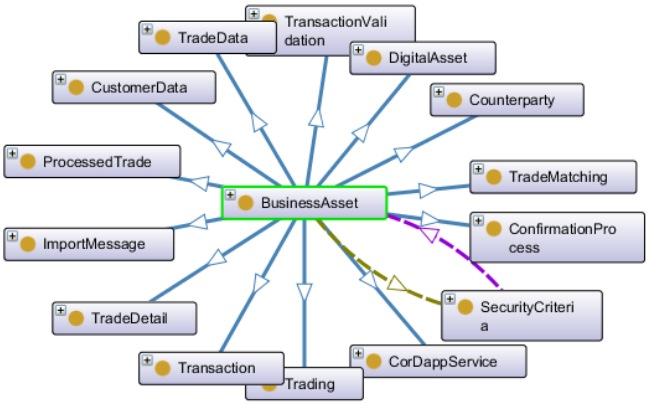

Fig. 13. Business assets classification

System assets at the same time could be business assets, it depends what asset value to protect from a security threat. For example, system assets "TradeDetail, Transaction and LogFile" support business asset "ProcessedTrade". Here, "TradeDetail" is an example of a business asset that is supported by system assets "Database, SenderEngine, MatchingEngine". The class definition of Asset is:

```
Class (Asset SubClass (
    BusinessAsset SystemAsset
) class BusinessAsset (
    restriction (hasConstraint someValuesFrom ( SecurityCriteria )
) class SystemAsset (
    restriction (supports someValuesFrom ( BusinessAsset )
)
```

Asset class has subclasses (e.g., BusinessAsset and SystemAsset) and a restriction "hasConstraint" on someValuesFrom the BusinessAsset. The someValuesFrom restriction presents that security criteria is not a constraint on all the business assets.

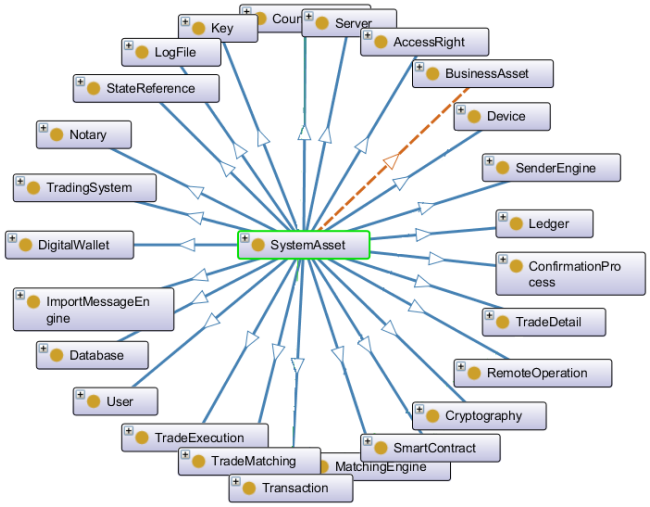

Fig. 14. System assets classification

4.2.2 Security threats: Security threats are classified as active or passive threats (Fig. 15). Security threats classification is built upon the threats that are mitigated and appear within CorDapp. For example, in financial industry, security threats related to spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege are mitigated by using a blockchain-based solution (e.g., CorDapp). In contrast, some other threats appear (e.g., endpoint threat, quantum computing threat, privacy violation, de-anonymization, smart contract threat and denial-of-state) in CorDapp that appear by using a Corda platform. Security threats exploit vulnerabilities and target some business asset(s). The DL for relation "exploits" and "targets" is:

```
exploits some Vulnerability / targets some BusinessAsset
```

Threat class has subclasses (e.g., ActiveThreat, PassiveThreat) and a restriction "exploits" on someValuesFrom the Vulnerability. Another restriction "targets" on someValuesFrom the SystemAsset. The someValuesFrom restriction illustrates that a particular threat exploits particular vulnerabilities fully or partially and targets SystemAsset.

```
Class (Threat SubClass (
        ActiveThreat PassiveThreat
    ) restriction (
        exploits someValuesFrom ( Vulnerability )
    ) restriction (
        targets someValuesFrom ( SystemAsset )
    )
)
```

For example, in CorDapp a malicious actor is able to create a transaction and non-validating notary consumes a state by considering it a valid transaction. The "DenialOf-

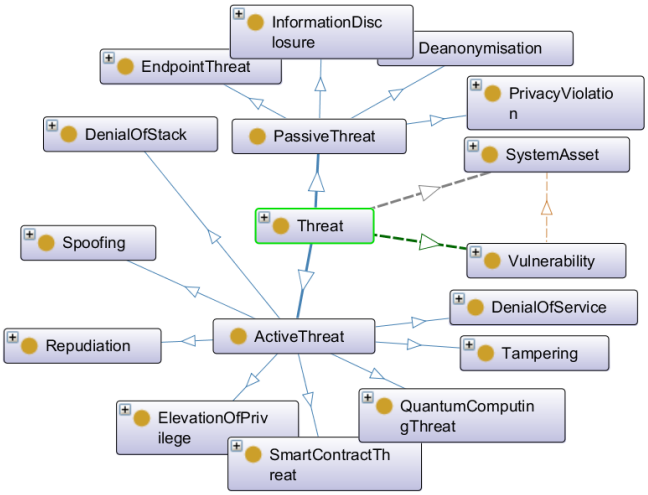

Fig. 15. Security threats classification

Stack" threat exploits a vulnerability "NonValidatingNotariesConsumeState" within CorDapp when non-validating notaries consume state. The "DenialOfStack" threat targets both "NonValidatingNotary" and "StateReference" SystemAsset.

4.2.3 Vulnerabilities: Vulnerabilities classification (Fig. 16) is built by identifying the weakness within the system that enables a particular security threat. A vulnerability is a characteristic of system asset(s) and negates the security criteria of a business asset. The DL for relation "characteristicOf" and "negates" is:

```
characteristicOf some SystemAsset / negates some SecurityCriteria
```

Vulnerability class definition explains that it contains various vulnerabilities that are characteristic of system assets and negates the security criteria of business assets.

```
Class (Vulnerability SubClass (
        MissingAccessControl
        ErrorProneSmartContract
        .....
    ) restriction (
        negates someValuesFrom ( SecurityCriteria )
    ) restriction (
        characteristicOf someValuesFrom ( SystemAsset )
    )
)
```

For example, the missing or improper implementation of access control presents a weakness within a system. A malicious actor could exploit this vulnerability and get

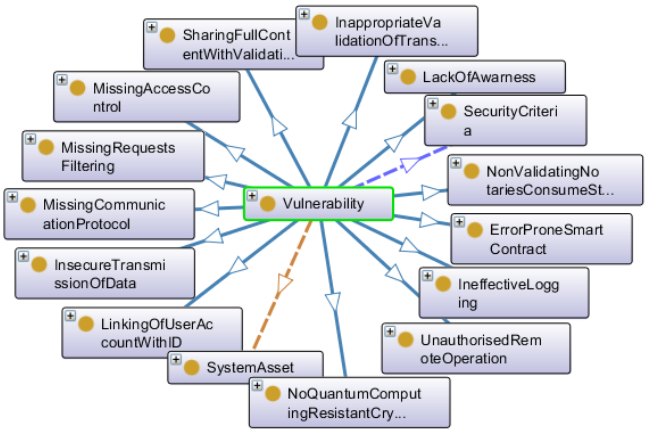

Fig. 16. Vulnerabilities classification

unauthorised access in the system. A vulnerability "MissingAccessControl" is a characteristicOf "SystemAsset" (ConfirmationProcess, Database, ImportMessageEngine, LogFile, MatchingEngine, SenderEngine, TradeDetail, TradeMatching and Transaction) and negates some (Availability or Confidentiality or Integrity).

4.2.4 Countermeasures: Countermeasures classification (Fig. 17) presents the counteract that implements security requirements. In contrast to security requirements, countermeasures implement the security requirements to mitigate security threats and improve the security of the system. The DL for relation "implements" is:

```
implements some SecurityRequirement
```

Countermeasure class definition explains that it contains various countermeasures that implement the security requirements.

```
Class (Countermeasure  SubClass (
        AccessControl
        SecureCommunication
        .....
    ) restriction (
        implements someValuesFrom ( SecurityRequirement )
    )
)
```

For example, countermeasure "AccessControl" implements "RestrictRemoteOperation" and "RestrictUnauthorisedAccess" security requirements.

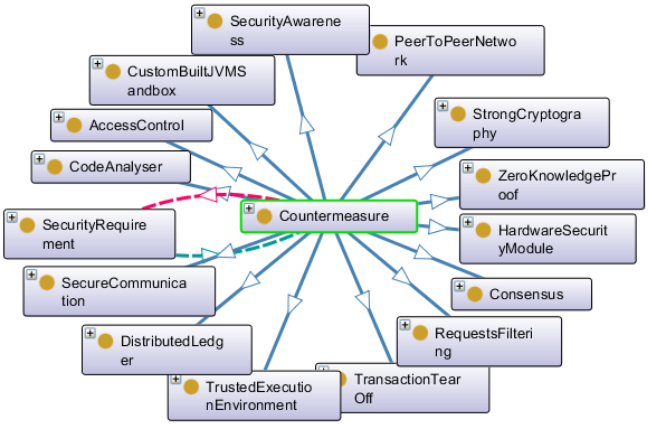

Fig. 17. Countermeasures classification

5 Ontology evaluation

We follow the task-based (Raad and Cruz, 2018) ontology evaluation approach. This approach helps us to distinguish how CordaSecOnt improves the SRM of CorDapp.

5.1 CorDapp to mitigate security threats

In this section, we perform a SRM of post-trade matching and confirmation using CordaSecOnt. Firstly, we organise the business assets, their security criteria, system assets that support business assets, and what security threats target system assets (Table 6). Secondly, we present the vulnerabilities in a centralised infrastructure of post-trade matching and confirmation that exploit by different security threats. Finally, we illustrate the security controls to mitigate security threats.

Table 6. Assets to be secured in post-trade matching and confirmation

Business asset	**System asset**	**Threat**
Import message *(I)*	Sender engine, Import message engine	*S*
Trade data *(I)*	Database, Trade matching, Confirmation process	*T, R*
Trade detail *(I, A)*	Database, Sender engine, Matching engine	*T, D*
Processed trade *(C, I)*	Trade detail, Transaction, Log file	*Id, R*
Trade matching *(I, A)*	Server, Import message engine, Matching engine	*Id, D*
Confirmation process *(A)*	Server, Trading system, Sender engine	*D*
Trading *(I)*	Access right, Remote operation, Trade execution	*E*

The architecture (Fig. 18) helps to visualise the vulnerable assets in centralised post-trade matching and confirmation. The vulnerabilities (Table 7) depict weaknesses of

Table 7. Vulnerabilities that exploit by security threats

	Vulnerability	**Characteristic of**	**Threat**
V#1	Missing communication protocol	Import message engine, Sender engine	*S*
V#2	Insecure transmission of data	Confirmation process, Database, Import message, Matching & Sender engine, Trade matching	*S, T*
V#3	Inappropriate validation of transmitted data	Confirmation process, Database, Matching & Sender engine, Trade matching	*T*
V#4	Missing access control	Confirmation process, Database, Log file, Import message, Matching & Sender engine, Trade detail, Trade matching, Transaction	*S, T, R*
V#5	Ineffective logging	Confirmation process, Database, Log file, Import message & Matching engine, Trade detail, Trade matching, Transaction, Server	*R, Id*
V#6	Missing requests filtering	Database, Import message, Matching & Sender engine, Trading system, Server	*D*
V#7	Unauthorised remote operation	Access right, Remote operation, Trade execution	*E*

the system assets and exploit by various security threats. The V#1 belongs to missing or no proper implementation of communication protocols and it exploits by spoofing threat (Brubaker et al., 2014), (POV Network, 2017). The in-secure transmission of data (V#2) could be exploited by spoofing or tampering threats (Polyviou et al., 2019), (Maurer et al., 2017). The inappropriate validation of transmitted data (V#3) enables tampering threat (AL-essa, 2019), (Accenture Security, 2019). The poorly implemented or having no access control (V#4) could enable spoofing, tampering and repudiation threats (Maurer et al., 2017), (Accenture Security, 2019). The ineffective logging (V#5) (Maurer et al., 2017), (Accenture Security, 2019) could happen because of "insignificant log message", "logging sensitive information", "unprotected logs", "centralised control on logs" or having "no backup of logs". The V#5 could exploit by repudiation and information disclosure threats. If a system has no proper mechanism to filter a large number of requests (V#6) (WEB, a), (Maurer et al., 2017) then denial of service threat could exploit V#6. The V#7 could exploit by elevation of privilege threat if a system has weak controls to restrict unauthorised remote operations (V#7) (Maurer et al., 2017), (Accenture Security, 2019).

We present *CorDapp as a countermeasure solution* (Fig. 11). The CorDapp countermeasures (Table 8) implements security requirements that mitigate security threats and protect the assets. The architecture (Fig. 11) illustrates the assets of post-trade matching and confirmation and presents the CorDapp-based countermeasures (CC). To mitigate *V#1*, CorDapp considers only authorised nodes over a P2P network where nodes behave both as client and server (Dagan et al., 2018). Also, the CorDapp incorporates a mutually authenticated TLS connection (Corda Threat Model, 2018) to protect the communication between nodes.

The in-secure transmission of data could lead to spoofing and tampering threats *(V#2)* (Polyviou et al., 2019), (Maurer et al., 2017). The malicious user may intercept the plain-text data transmission and gain unauthorised access to the system. It would negate the confidentiality and integrity of data. The CorDapp mitigates by incorporating

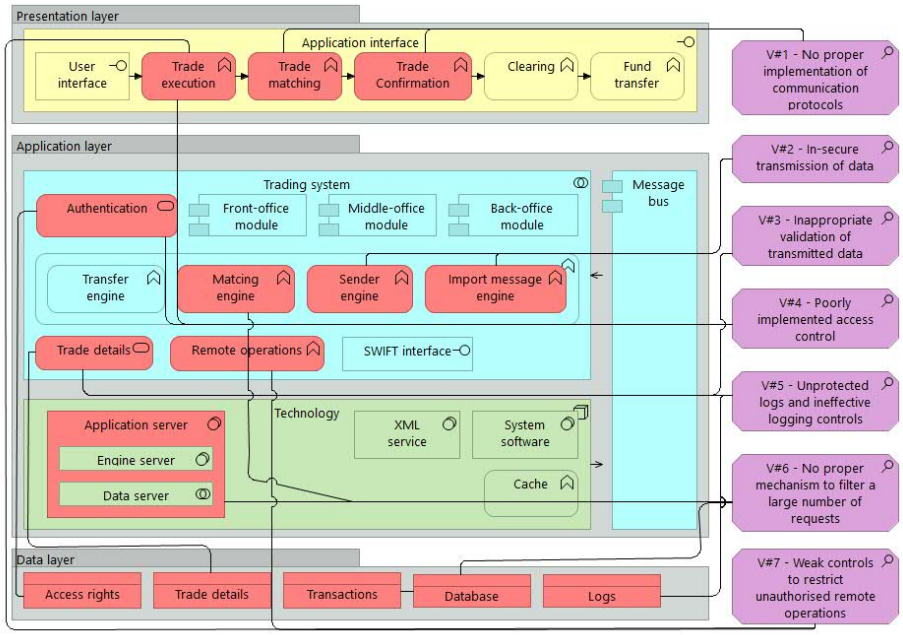

Fig. 18. Security threats vulnerabilities in centralised infrastructure of post-trade

PKI based cryptography and Intel SGX integration (Corda Threat Model, 2018) that would bring the CPU P2P encryption and allows one to encrypt the entire ledger (Hearn, 2016). Also, a hardware security module (HSM) could be applied to manage and protect digital keys (Corda Threat Model, 2018).

To mitigate *V#3*, the transaction validation mechanism is utilised. The Corda platform uses a notaries-based decentralised consensus model to validate a transaction and ensure authenticity and integrity (Corda Threat Model, 2018), (Hearn, 2016). In a centralised approach, the attacker can trigger a Man-in-the-Middle (MitM) attack and modify the transaction (Kubo et al., 2018). Similarly, in CorDapp the attacker can perform MitM to modify the transaction but the notaries-based consensus model protects and guarantee the integrity of transaction (Corda Threat Model, 2018). The *V#4* is mitigated by a decentralised access-control in CorDapp. Also, only authorised nodes can join the network that limits this vulnerability.

Multi-user applications are subjects to repudiation because the system allows a user to perform/deny the malformed actions. The system should ensure that the user actions are recorded in order to protect against insider security risks. In order to mitigate *V#5*, CorDapp manages the records in a decentralised immutable ledger. It provides tamper-proof transparent traceability and auditing. Also, the CorDapp logs each action of a participant node that replicates over a P2P network (Corda Threat Model, 2018).

The *V#6* is mitigated by introducing requests rate-limiting firewall. In CorDapp the P2P communication is authenticated as a part of the TLS protocol *(CC#1, CC#2)*;

Table 8. Countermeasures that implement security requirements to mitigate threats

	Countermeasure	**Implements**	**Mitigate**
CC#1	P2P network & Authorised nodes	Filter large number of request, Incorporate secure data transmission	*S, T, D*
CC#2	Mutually-authenticated TLS connection & PKI	Implement communication protocol, Make data unreadable before transmission	*S*
CC#3	Consensus	Validate transmitted data	*T*
CC#4	Access control	Restrict remote operation, Restrict unauthorised access	*S, T, R, E*
CC#5	Distributed ledger	Enable appropriate logging, Protect logs	*R, Id*
CC#6	Firewall	Filter large number of request	*D*
CC#7	JVM sandbox	Restrict remote operation	*E*

it means that the attacker could not join the Corda network to launch a DoS attack (Corda Threat Model, 2018). To protect against unauthorised remote operations *(V#7)*, the CorDapp utilise the secure communication protocols *(CC#1)* along with the concept of custom-built JVM sandbox (Corda Threat Model, 2018) to prevent unauthorised remote operations and execution of code.

5.2 CorDapp security threats

In this section, we perform the SRM to mitigate security threats that appear within CorDapp. Similarly, first, we identify the business assets, system assets, security criteria and security threats (Table 9). For example, the i) endpoint vulnerability (EV) (such as keys lost, weak passwords, physical access to digital wallets and devices), ii) quantum computing threat (QCT), iii) privacy violation (PV), iv) de-anonymization (DA), v) smart contract attack (SCA), and vi) denial-of-state attack (DSA).

Table 9. CorDapp-based post-trade assets and security threats

Business assets	**System assets**	**Threat**
CorDapp service *(I, A)*	Digital wallet, Keys, Computers/devices, User	*EV*
Transaction *(I)*	Trade detail, Cryptography	*QCT*
Customer data *(C)*	Trade detail, Counter-parties	*PV*
Counter-parties *(C)*	Transaction, Trade detail, Counter-parties	*DA*
Digital asset *(I)*	Smart contract, Ledger	*SCA*
Transaction validation *(I, C)*	Non-validating notary, State reference	*DSA*

The vulnerabilities (Table 10) belong to post-trade matching and confirmation in CorDapp, called CorDapp vulnerabilities (CV). The vulnerabilities are mapped on the architecture (Fig. 19) to visualise which assets are affected by these vulnerabilities.

The lack of awareness and knowledge *(CV#1)* about security could trigger the endpoint vulnerability (Velissarios et al., 2019). For example, if an attacker learns about the private key then he can utilise to acquire access and ownership to data. Quantum computing research is emerging and advancing in modern technology. In Blockchain,

not using quantum-resistant cryptography *(CV#2)* could put consensus mechanisms and private keys are at high risk in a post-quantum era (Velissarios et al., 2019). In CorDapp, validating notary observe the full content of the transaction to validate it and Sharing full content of transaction with validating notaries *(CV#3)* could lead to privacy violation (Koens et al., 2019). In CorDapp possible to link individuals data that could trigger de-anonymization threat (Moser, 2017), (Koens et al., 2019). Error-prone smart contracts (Corda Secure Coding Guidelines, 2020) *(CV#5)*, for example, a smart contract could have a logical bug, missing error handling, missing input validation or misuse of programming language construct. In CorDapp, the malicious actor is able to create a transaction and non-validating notary consumes a state (Koens et al., 2019) by considering it a valid transaction *(CV#6)*.

Table 10. Vulnerabilities that appear within CorDapp and exploit by security threats

	Vulnerability	**Characteristic of**	**Threat**
CV#1	Lack of awareness	Device, Digital wallet, Keys, User	*EV*
CV#2	No QC resistant cryptography	Cryptography, Trade detail	*QCT*
CV#3	Sharing content with validating notaries	Counter-party, Trade detail	*PV*
CV#4	Linking of user account with ID	Counter-party, Trade detail, Transaction	*DA*
CV#5	Error prone smart contract	Ledger, smart contract	*SCA*
CV#6	Non-validating notaries consume state	Non-validating notary, state reference	*DSA*

We collect various countermeasures (Table 11) that implement the security requirements and overcome these vulnerabilities. The architecture (Fig. 20) illustrates how the countermeasures for CorDapp vulnerabilities (CCV) are applied to secure the CorDapp.

Table 11. Countermeasures that implement security requirements to mitigate threats of CorDapp

	Countermeasure	**Implements**	**Mitigate**
CCV#1	Security awareness	Improve security awareness & knowledge	*EV*
CCV#2	Hardware security module (HSM)	Incorporate secure data transmission, Protect user digital possession	*EV*
CCV#3	Quantum-resistant cryptography	Use Quantum computing resistant cryptography	*QCT*
CCV#4	Transaction tear-off	Protect transaction content from validating notary	*PV, DA*
CCV#5	Code analyser	Perform code analysis	*SCA*
CCV#6	Trusted execution environment Zero-knowledge proof	Incorporate secure data transmission, Protect transaction state from non-alidating notary, Restrict linking of individual data and account	*DSA*

The lack of knowledge and awareness *(CV#1)* led attackers to steal information (Bellekens et al., 2016) by social engineering and phishing (WEB, a), or accidentally exposing the secure information (Maurer et al., 2017). The *CCV#1* (Velissarios et al., 2019) is related to educate system users about possible security risks if exposing their

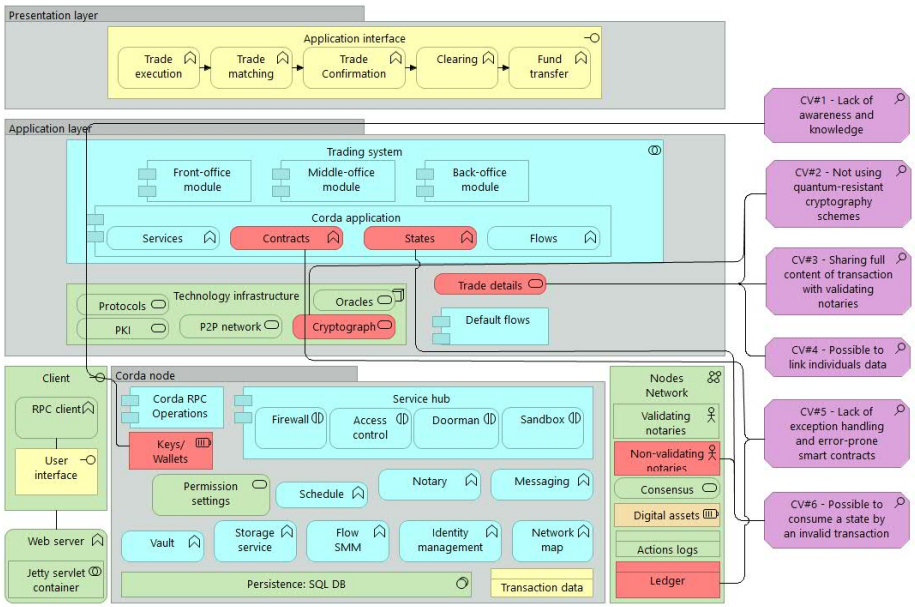

Fig. 19. Security threats vulnerabilities that appear within CorDapp

protected information. The organisations should arrange staff security training's and establish a disciplinary process, promote incident reporting culture within an organisation and imply user security policies. Also, incorporate hardware security modules (HSM) *(e.g., AWS cloud HSM, Azure key vault, Futurex, GemaltoLuna, N-cipher N-shield, Securosys Primus X or Utimaco)* to generate, protect, and store keys *(CCV#2)* (Corda Threat Model, 2018), (Velissarios et al., 2019).

The quantum computing threat (Yin et al., 2018) is real, however, CorDapp does not provide any mechanism *(CV#2)* to tackle this threat in a post-quantum era. The possible way is to implement quantum-resistant cryptography schemes *(CCV#3) (e.g., lattice-based, multivariate, hash-based, code-based, symmetric key quantum resistance and supersingular elliptic curve isogeny cryptography)* to secure against quantum computing threats (Yin et al., 2018), (Koens et al., 2019).

The Koens et al., (2019) suggested to use transaction tear-off *(CCV#4)* to protect against *CV#3* and *CV#4*. For example, this concept within CorDapp would increase privacy, because it tear-off the information and shows a minimum amount of information that should be kept confidential from the transaction (Transaction Tear-Offs, 2020).

The lack of exception handling and error-prone smart contracts *(CV#5)* could lead to harm valuable assets, for example, Ethereum smart contract reentrancy attack when an adversary stole $60 million Ethers (Atzei et al., 2017). The system should include a smart contract's code analyser *(CCV#5)* (Atzei et al., 2017) to detect errors, identify race conditions and sanitise the smart contract code. In *CCV#6*, the author (Koens et al.,

2019) suggested to use trusted execution environments (TEE), for example, IntelSGX and zero-knowledge proof (ZKP) to protect against *CV#6*.

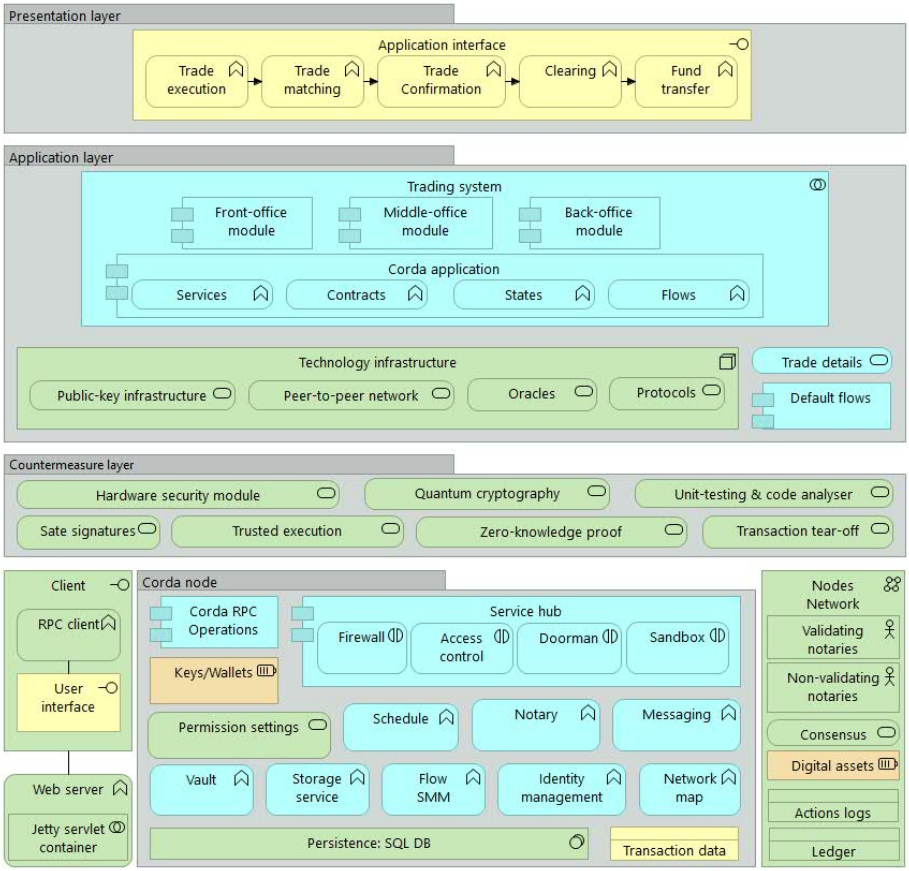

Fig. 20. Countermeasures to mitigate CorDapp security threats [need redesign]

6 Related work

At the beginning of ontology development, we explore various literature that builds the ontologies to formalise and structure the domain of information security (Table 12). Herzog et al., (2007) present OWL-based ontology in the domain of information security. It is a generic information security ontology that includes the core concepts of assets, threats, vulnerabilities, countermeasures and their relations. The authors describe the ontology hierarchy and reasoning capabilities by classifying assets, threats, vulnerabilities and countermeasures in taxonomical structures. The ontology is created using the Protege OWL tool and SPARQL to retrieve data from ontology.

Fenz et al., (2009) provide an ontological structure for information security domain knowledge. The ontology model the concepts of assets, threats, vulnerabilities and con-

trols and their implementation. These concepts grouped into three sub-ontologies (security, enterprise and location). The ontology follows the OWL-DL to define relations between the concepts of sub-ontologies. The authors use the fire threat example to ensure that ontology is generic enough to cover the entire information security domain and supports the SRM process.

Security requirements are difficult to elicit, analyse, and manage (Souag et al., 2015), the authors (Souag et al., 2015) use the ontological approach to facilitate security requirements elicitation process. The ontology uses the concepts of organisation dimensions (e.g., assets), risk dimensions (e.g., risk, threats, vulnerabilities and impact), treatment dimension (e.g., security requirements). The ontology is developed using OWL, Protégé and Semantic Query-Enhanced Web Rule Language (SQWRL) to query data from ontology. The authors performed the controlled experiment to demonstrate that the ontology helps in security requirements eliciting process.

Table 12. Mapping of security ontologies to the CordaSecOnt. The acronym defined in the table are: *Se. C. - Security Criteria, Vul. - Vulnerability, RT - Risk Treatments, SR - Security Requirements, Cou. - Countermeasures, DM - SRM Domain Model, BC - Blockchain.*

	Asset	**Se. C.**	**Threat**	**Vul.**	**RT**	**SR**	**Cou.**	**DM**	**BC**
Herzog et al., (2007)	x	–	x	x	x	–	x	–	–
Fenz et al., (2009)	x	–	x	x	–	–	x	–	–
Souag et al., (2015)	x	–	x	x	x	x	x	–	–
Vega-Barbas et al., (2019)	x	–	x	x	–	–	x	–	–
Mozzaquatro et al., (2018)	x	–	x	x	x	–	x	–	–
Zamfira et al., (2018)	–	–	x	–	–	–	x	–	–
Silva and Rafael (2017)	x	x	x	x	x	x	x	–	–
Gao et al., (2013)	x	–	x	x	x	–	x	–	–
CordaSecOnt	x	x	x	x	x	x	x	x	x

Substantial research has been conducted to formalise and structure the knowledge of information security. The above discussed related work represent the examples of generic information security ontologies, there also exist domain-specific ontologies to structure knowledge of information security for a specific domain. For example, Vega-Barbas et al., (2019) present an ontology-based system for dynamic risk management in administrative domains. The work collected and modelled the assets, threats and vulnerabilities in the administrative domain. The Mozzaquatro et al., (2018) develop an ontology-based cybersecurity framework for the Internet of Things (IoT) to improve the IoT cybersecurity focusing on the enterprise monitoring, analysis and classification of security vulnerabilities. The IoT cybersecurity ontology framework deals security of IoT at design time, run time and an integration layer. The ontology uses OWL and Protégé to classify assets, vulnerabilities, threats, security properties and security mechanisms in IoT security domain.

Zamfira et al., (2018) build the ontology of cyber-operations in networks of computers. The ontology improves the detection capabilities of attacks at various levels of application. Silva and Rafael (2017) present the ontologies for network security and

discuss future challenges. The (Gao et al., 2013) created an ontology-based security assessment framework of network and computer attacks. The ontology explains taxonomy of attacks in the context of attack impact, attack vector, attack target, vulnerability and defence strategies.

The related works so far develop either generic or domain-specific ontology of information security mainly focusing on the *centralised applications*. In the above work, the definition of assets is not concrete, for example, no categorisation of business and system assets. Also, missing the security criteria of assets. The relations between risk treatments-related concepts are not explicitly described, for example, what risk treatment decisions refine security requirements that implement by security controls (e.g., countermeasures). The relations between concepts are defined mainly based on assumptions and neglected the use of information security or SRM domain model.

In contrast, our ontology focusing on the information security of blockchain-based decentralised applications where we utilise the Corda platform and evaluate the ontology using post-trade matching and confirmation case of financial industry. Our ontology uses the SRM domain model relations that explicitly explains the assets, risks and risks treatments-related concepts. Also, SRM domain model provides the security-based technical vocabulary for our ontology to define concepts and relations.

7 Discussion and concluding remarks

In this work, we utilise the SRM domain model and STRIDE to build Corda-based security ontology for post-trade matching and confirmation. We define the scope of our ontology and develop the classifications related to assets, security criteria, threats, vulnerabilities, risk treatments, security requirements and countermeasures. Later, we evaluate the ontology by performing SRM of capital markets post-trade matching and confirmation case. In SRM, we explore the security threats in two different perspectives: (*i*) the security threats that are mitigated *(e.g., spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege)* by using CorDapp, and (*ii*) security threats that appear *(e.g., endpoint threat, quantum computing threat, privacy violation, de-anonymization, smart contract threat and denial-of-state)* within CorDapp by using a blockchain-based Corda platform.

CordaSecOnt is a publicly available knowledge base of information security that combines blockchain-based Corda platform. The CordaSecOnt could support the developers' to perform SRM while developing financial industry CorDapps. Also, the CordaSecOnt encode the static knowledge of Corda information security to dynamic ontology-based knowledge that could be extended, reuse or integrate with other security ontologies. The CordaSecOnt is the first blockchain-based information security ontology and this work could trigger the development of more detailed and acceptable ontology representation in the domain of blockchain-based information security.

As a part of future work, we would like to validate the Corda security ontology by evaluating it with information security experts. The experts would provide a discussion on a question *"Is CordaSecOnt supports a process of SRM when building financial industry CorDapps"?*. In another future work, we would utilise the findings of this research to build an ontology-based security risk reference model for blockchain-based

applications to evaluate their security. The identified components of CorDapp would be generalised in a way that would not be dependent on the specific blockchain type or platform. The ontology-based security risk reference model would explain and help to explore the blockchain-based applications in the direction of assets-related, risks-related and risks treatments-related concepts.

Acknowledgement. The authors would like to thank Justs Placāns (Riga Technical University) for the constructive comments and significant contribution while preparing this paper.

References

Accenture Security. (2019). Future Cyber Threats : Extreme But Plausible Threat Scenarios In Financial Services, 32.

Al-essa, M. (2019). The Impact of Blockchain Technology on Financial Technology (FinTech)

Ali, S., Wang, G., White, B., Cottrell, R. L. (2018). A Blockchain-Based Decentralized Data Storage and Access Framework for PingER. *Proceedings of 17th IEEE Trustcom and 12th IEEE BigDataSE 2018*, 1303–1308.

Atzei, N., Bartoletti, M., Cimoli, T. (2017). A survey of attacks on Ethereum smart contracts (SoK).

Baker, R. P. (2015). The Trade Lifecycle: Behind the Scenes of the Trading Process (2nd ed.).

Bellekens, X., Hamilton, A., Seeam, P., Nieradzinska, K., Franssen, Q., Seeam, A. (2016). Pervasive eHealth services a security and privacy risk awareness survey. *Proceedings of International CyberSA (2016)*, 1–4.

Brubaker, C., Jana, S., Ray, B., Khurshid, S., Shmatikov, V. (2014). Using frankencerts for automated adversarial testing of certificate validation in SSL/TLS implementations. *Proceedings of IEEE Symposium on Security and Privacy*, 114–129.

Buterin, V. (2014). A Next-Generation Smart Contract and Decentralized Application Platform. Retrieved from `https://github.com/ethereum/wiki/wiki/White-Paper`

Corda Deterministic JVM, (2020) `https://docs.corda.net/docs/corda-os/4.6/key-concepts-djvm.html`

Corda Threat Model. `https://github.com/corda/corda/pull/3817/files/7a89ee87259dc6c2fd2594372be1050d3cf07009`

Dagan, G. (2018). The Actual Networking behind the Ethereum Network: How It Works. `https://medium.com/orbs-network/the-actual-networking-behind-the-ethereum-network-how-it-works-6e147ca36b45`

Dubois, É., Heymans, P., Mayer, N., Matulevičius, R. (2010). A Systematic Approach to Define the Domain of Information System Security Risk Management. *Proceedings of Intentional Perspectives on Information Systems Engineering)*, 289-306.

ECB (2017). The potential impact of DLTs on securities post trading harmonisation and on the wider EU financial market integration. `http://finadium.com/ecb-potential-impact-of-dlts-on-securities-post-trading-harmonization-and-on-the-wider-eu-financial-market-integration`

Fenz, S., Ekelhart, A. (2009). Formalizing information security knowledge. *Proceedings of the 4th International Symposium on ACM Symposium on Information, Computer and Communications Security, ASIACCS'09*, 183–194.

Gao, J. B., Zhang, B. W., Chen, X. H., Luo, Z. (2013). Ontology-based model of network and computer attacks for security assessment. *Proceedings of Journal of Shanghai Jiaotong University (Science), 18(5)*, 554–562.

Hearn, M. (2016). Corda: A distributed ledger (Whitepaper), 1–56. https://www.corda.net/content/corda-technical-whitepaper.pdf

Hector, U.-R., Boris, C.-L. (2020). BLONDiE: Blockchain Ontology with Dynamic Extensibility.

Herzog, A., Shahmehri, N., Duma, C. (2007). An Ontology of Information Security. *Proceedings of International Journal of Information Security and Privacy (IJISP), 1(4)*, 1–23.

Iqbal, M., Matulevičius, R. (2019). Blockchain-Based Application Security Risks: A Systematic Literature Review. *Proceedings of CAiSE 2019 Advanced Information Systems Engineering Workshops*, 1–26.

Iqbal, M., Matulevičius, R. (2019). Comparison of Blockchain-Based Solutions to Mitigate Data Tampering Security Risk. *Proceedings of BPM 2019 International Conference on Business Process Management Blockchain Forum and CEE Forum*, 13-28.

Iqbal M., Matulevičius R. (2020). Managing Security Risks in Post-Trade Matching and Confirmation Using CorDapp. *Proceedings of DB&IS 2020 Databases and Information Systems*, 325-339.

Koens, T., King, S., Bos, M. Van Den, Wijk, C. Van, Koren, A. (n.d.). Solutions for the Corda Security and Privacy Trade-off : Having Your Cake and Eating It.

Kubo, R. (2018). Detection and Mitigation of False Data Injection Attacks for Secure Interactive Networked Control Systems. *Proceedings of 2018 IEEE International Conference on Intelligence and Safety for Robotics (ISR)*, 7–12.

Liu, C., Liu, H., Cao, Z., Chen, Z., Chen, B., Roscoe, B. (2018). ReGuard: Finding reentrancy bugs in smart contracts. *Proceedings of International Conference on Software Engineering*, 65–68.

Macrinici, D., Cartofeanu, C., Gao, S. (2018). Smart Contract Applications within Blockchain Technology: A Systematic Mapping Study. *Proceedings of Telematics and Informatics, (October)*, 0–1.

Matulevičius, R. (2017). Fundamentals of secure system modelling. Springer.

Maurer, T., Levite, A., George. P. (2017). Toward a global norm against manipulating the integrity of financial data. *Proceedings of Economics Discussion Papers, No. 2017-38* 1–42.

Millier, E. (1998). An Introduction to the Resource Description Framework. *Proceedings of Bulletin of the American Society for Information Science*.

Moser, J. (2017). The Application and Impact of the European General Data Protection Regulation on Blockchains. https://www.r3.com/wp-content/uploads/2018/04/GDPR_Blockchains_R3.pdf

Mozzaquatro, B. A., Jardim-Goncalves, R., Agostinho, C. (2015). Towards a reference ontology for security in the Internet of Things. *Proceedings of 2015 IEEE International Workshop on Measurements and Networking*, 117–122.

Mozzaquatro, B. A., Melo, R., Agostinho, C., Jardim-Goncalves, R. (2016). An ontology-based security framework for decision-making in industrial systems. *Proceedings of 4th International Conference on Model-Driven Engineering and Software Development)*, 779–788.

Mozzaquatro, B. A., Agostinho, C., Goncalves, D., Martins, J., Jardim-Goncalves, R. (2018). An ontology-based cybersecurity framework for the internet of things. *Proceedings of Sensors (Switzerland), 18(9)*, 1–20.

Noy, N. F., McGuinness, D. L. (2001). Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory, (January 2001).

OWL Working Group, (2012). Web Ontology Language (OWL). https://www.w3.org/OWL/

Parag (2020). Transaction flow in R3 Corda. https://blockchainhlce.com/transaction-flow-in-r3-corda/

Placāns, J. (2019). Security Risk Management in Corda-based Application for Capital Markets.

Polyviou, A., Soldatos, P. V. and J. (2019). Blockchain Technology: Financial Sector Applications Beyond Cryptocurrencies, *Proceedings pf MDPI 2019 28*, 1–5.

POV Network (2017). Proof of Authority: consensus model with Identity at Stake. https://medium.com/poa-network/proof-of-authority-consensus-model-with-identity-at-stake-d5bd15463256

Pradeepkumar, D. S., Singi, K., Kaulgud, V., Podder, S. (2018). Evaluating complexity and digitizability of regulations and contracts for a blockchain application design. *Proceedings of 2018 ACM/IEEE 1st International Workshop on Emerging Trends in Software Engineering for Blockchain*, 25–29.

Raad, J., Cruz, C. (2018). A Survey on Ontology Evaluation Methods. textitHAL Archives-Ouvertes.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublauch, H., Stevens, R., Wang, H., Wroe, C. (2004). OWL pizzas: Practical experience of teaching OWL-DL: Common errors and common patterns. *Proceedings of Lecture Notes in Artificial Intelligence*, 3257, 63–81.

Ruffy, F., Hommel, W., Eye, F. Von. (2016). A STRIDE-based Security Architecture for Software-Defined Networking. *Proceedings of ICN 2016: The Fifteenth International Conference on Networks*, 95–101.

Silva, D. V., Rafael, G. R. (2017). Ontologies for network security and future challenges. *Proceedings of the 12th International Conference on Cyber Warfare and Security, ICCWS 2017*, 541–547.

Skuce, D. (1995). Conventions for reaching agreement on shared ontologies. *Proceedings of the 9th Knowledge Acquisition for Knowledge Based Systems Workshop (1995)*.

Souag, A., Salinesi, C., Mazo, R., Comyn-Wattiau, I. (2015). A security ontology for security requirements elicitation. *Proceedings of Lecture Notes in Computer Science*, 157–177.

Transaction Tear-Offs (2020). https://docs.corda.net/docs/corda-os/4.6/key-concepts-tearoffs.html

Uschold, M., Gruninger, M. (1996). Ontologies : Principles , methods and applications Ontologies : Principles , Methods and Applications. *Proceedings of Knowledge Engineering Review, 1996*, 11(2).

Vega-Barbas, M., Villagrá, V. A., Monje, F., Riesco, R., Larriva-Novo, X., Berrocal, J. (2019). Ontology-based system for dynamic risk management in administrative domains. *Proceedings of Applied Sciences (Switzerland), 9(21)*.

Velissarios, J., Herzig, J., Unal, D. (2019). Blockchain's potential starts with security. *Accenture*, 1–20.

Yin, W., Wen, Q., Li, W., Zhang, H., Jin, Z. (2018). An Anti-Quantum Transaction Authentication Approach in Blockchain, *Proceedings of IEEE Access Journal*

Zamfira, A. C., Ciocarlie, H. (2018). Developing an ontology of cyber-operations in networks of computers. *Proceedings of 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing, ICCP 2018*, 395–400.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., Wang, H. (2016). Blockchain Challenges and Opportunities : A Survey Shaoan Xie Hong-Ning Dai Huaimin Wang. *Proceedings of International Journal of Web and Grid Services, 14(4)*, 1–24.

WEB a. Cybersecurity essentials for capital markets firms in the digital age. https://www.wipro.com/capital-markets/cybersecurity-essentials-for-capital-markets-firms-in-the-digital-age

WEB b. Proof of Stake FAQs. https://eth.wiki/en/concepts/proof-of-stake-faqs

WEB c. Corda nodes. https://docs.corda.net/docs/corda-os/4.6/key-concepts-node.html

WEB d. Corda architecture overview. https://solutions.corda.net/deployment/onprem/corda-node-architecture-components.html

WEB e. Ontology. https://www.oxfordlearnersdictionaries.com/definition/english/ontology

Appendix

Classifications

Security criteria: Security criteria classification (Fig. 21) is based on the concepts of CIA triad. Security criteria represent the constraint of business assets. The relation "constraintOf" is an inverse of "hasConstraint". For example, security criteria "Integrity" is a constraint of business asset "ImportMessage", the same example is presented in Asset classification with the "hasConstraint" relation. The DL for relation "constraintOf" is:

```
constraintOf some BusinessAsset
```

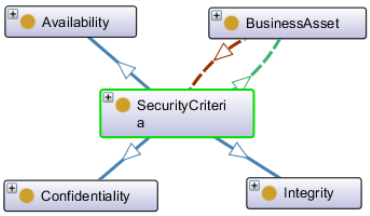

Fig. 21. Security criteria classification

If a security expert wants to explore security criteria of business assets within CorDapp, he can browse to CordaSecOnt to examine class definition. The class definition of SecurityCriteria is:

```
Class (SecurityCriteria SubClass (
        Confidentiality Availability Integrity
    ) restriction (
        constraintOf someValuesFrom ( BusinessAsset )
    )
)
```

The definition explains a class SecurityCriteria has subclasses (e.g., Confidentiality, Availability, Integrity) and a restriction "constraintOf" on someValuesFrom the BusinessAsset. The someValuesFrom restriction presents that security criteria is not a constraint of all the business assets.

Risk treatments: Risk treatments classification (Fig. 22) includes four different risk treatment decisions. Risk treatment enables a decision process to treat identified security threats. A treatment decision satisfies the security need and leads to security requirements. The DL for relation "leadsTo" is:

```
leadsTo some SecurityRequirement
```

RiskTreatment class definition explains that it has subclasses (e.g., Avoidance, Reduction, Retention and Transfer) that leads to refine security requirements.

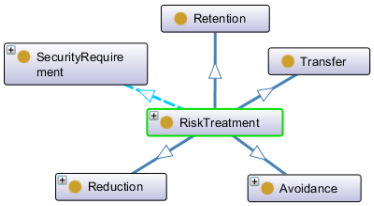

Fig. 22. Risk treatments classification

```
Class (RiskTreatment SubClass (
        Avoidance Reduction Retention Transfer
    ) restriction (
        leadsTo someValuesFrom ( SecurityRequirement )
    )
)
```

For example, risk treatment “Avoidance” leadsTo the refinement of security requirement “EnableAppropriateLogging, FilterLargeNumberOfRequest, IncorporateSecureDataTranmission and ValidateTransmittedData”.

Security requirements: Security requirements classification (Fig. 23) includes conditions to make true within the system to mitigate security threats. For example, a security requirement is the refinement of a risk treatment decision to mitigate the threats. The DL for relation “mitigates” is:

```
mitigates some Threat / mitigates some DenialOfStack
```

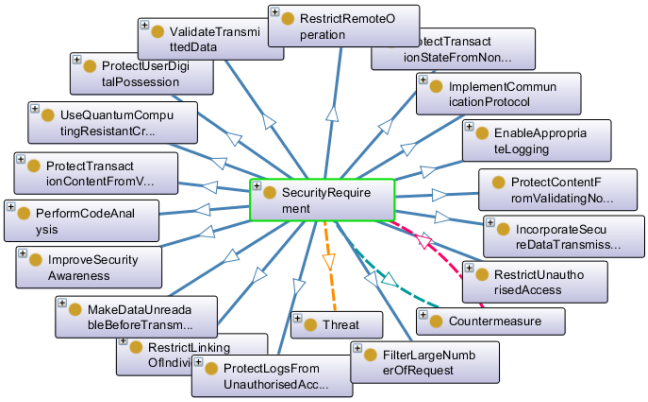

Fig. 23. Security requirements classification

SecurityRequirement class definition explains that it contains various security requirements to mitigate security threats.

```
Class (SecurityRequirement  SubClass (
        RestrictUnauthorisedAccess
        IncorporateSecureDataTransmission
        .....
    ) restriction (
        mitigates someValuesFrom (
            Threat (
                restriction (
                    unionOf (ActiveThreat PassiveThreat)
                )
            )
        )
    )
)
```

For example, security requirement "RestrictUnauthorisedAccess" mitigates Repudiation, Spoofing and Tampering threats.

Individuals

Individuals within OWL-based ontology represent instances of a class that share common characteristics. Individual instances are specific concepts that sometimes indicate the lowest level of granularity in the knowledge domain (Noy and McGuinness, 2001).

Class: Confidentiality - Individuals: Privacy, Secrecy
Confidentiality describes that information is not available or disclosed to unauthorised individuals, entities or processes (Matulevičius, 2017). Privacy and Secrecy both are instances of Confidentiality class because they share common characteristics. For example, keeping CustomerData secret to protect user privacy.

Class: Integrity - Individuals: Accuracy, Completeness
Integrity describes information authenticity and is not modified by an unauthorised individual (Matulevičius, 2017). Accuracy and Completeness both are instances of Integrity class because integrity looks after the accuracy and completeness of business assets. For example, TradeData should be integral by making sure the accuracy and completeness of the TradeData.

Class: DistributedLedger - Individuals: Auditable, Decentralised, Immutability, Provenance/Traceability, Replication/Redundancy, TamperProof, Transparent
A distributed ledger is shared and synchronized across multiple untrusted nodes on the P2P network. A DistributedLedger on the blockchain-based Corda platform should ease the examination process of financial transactions (Auditable), no third-party control the operations (Decentralised), nobody can modify the transaction once recorded on the ledger (Immutability), trace the origin of transaction or asset (Provenance/Traceability), shared the full copy of the ledger on the network nodes (Replication/Redundancy), validate the ledger is not changed or modified (TamperProof), network participant nodes

can view the relevant transactional information (Transparent).

Class: Device - Individuals: Computer
The class Device could consist of any hardware device, for example, Computer.

Class: AccessControl - Individuals: Decentralised
The class AccessControl has an instance Decentralised that indicates on blockchain-based Corda platform no third-party should manage the user access controls.

Class: PeerToPeerNetwork - Individuals: Decentralised, Disintermediary, Transparent
The class PeerToPeerNetwork has instances that illustrate P2P network should be Decentralised, Disintermediary and Transparent.

Class: CustomBuiltJVMSandbox - Individuals: Deterministic
The Corda platform custom-built JVM sandbox should be deterministic (Corda, 2020). Hence, class CustomBuiltJVMSandbox has an instance of Deterministic.

Class: ZeroKnowledgeProof - Individuals: Completeness, Soundness, ZeroKnowledge
The zero-knowledge proof should fulfil the criteria of completeness, soundness and zero-knowledge. Hence, the class ZeroKnowledgeProof has three different instances: Completeness, Soundness and ZeroKnowledge.

Ontology documentation

Inadequate documentation is the main barriers to effective knowledge sharing and understanding the ontology (Skuce, 1995). In order to overcome this issue, we document all the important assumptions and concepts, for example, the classes and sub-classes defined in the ontology, relations, individuals and meta-ontology to clarify what ontology is about and to interpret the meaning of ontological claims. Also, we use the Protégé annotation properties to document the terms (e.g., for classes, relations and individuals) separately that we used to build our ontology.

Ontology usage guidelines

This section belongs to the usage of ontology that explains how to use, integrate or extend this ontology. The guidelines include the resources of CordaSecOnt (Table 13) and educate the users that are not familiar with OWL or OWL-based tools. Our ontology is created by using Protégé ontology editor and the ontology is available and accessible online. We use the OntoGraf Protégé plugin to generate classifications graphs and Pellet reasoner to validate the consistency of our ontology. We also use PyLODE[1] (Python Live OWL Documentation Environment) tool to make human-readable form of ontology that give intuitive look to understand the encoded concepts within ontology and OWLGrEd[2] ontology visualisation tool to present graphical look of CordaSecOnt. In

[1] https://github.com/rdflib/pyLODE
[2] http://owlgred.lumii.lv/

order to use CordaSecOnt, first, install Protégé and open ontology. Second, retrieve information from ontology using following SPARQL queries. The CordaSecOnt could be integrated with other security ontologies and use the blockchain-based Corda platform security concepts. The ontology is available publicly at GitHub and could be extended.

Table 13. CordaSecOnt resources

Resource	Resource URL
CordaSecOnt	https://mmisw.org/ont/~mubashar/CordaSecOnt
GitHub	https://github.com/mubashar-iqbal/corda-security-ontology
Protégé	https://protege.stanford.edu/
OntoGraf	https://protegewiki.stanford.edu/wiki/OntoGraf
Pellet Reasoner	https://protegewiki.stanford.edu/wiki/Using_Reasoners
PyLODE	https://mmisw.org/pylode?url=https://mmisw.org/ont/~mubashar/CordaSecOnt
OWLGrEd	http://owlgred.lumii.lv/online_visualization/ln9o

SPARQL queries

The SPARQL queries can be used to retrieve information from an ontology. The following header code will remain the same for all the queries listed in this section.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX CordaSecOnt: <https://mmisw.org/ont/~mubashar/CordaSecOnt#>
```

System assets: Retrieve system assets that support business assets.

```
SELECT DISTINCT ?System_Asset ?Supports_Business_Asset  WHERE {
    ?System_Asset rdfs:subClassOf CordaSecOnt:SystemAsset .
    ?System_Asset rdfs:subClassOf ?Supports_Business_Asset .
    ?Supports_Business_Asset owl:onProperty CordaSecOnt:supports .
}
```

Business assets: Use the following query to get the business assets that have security criteria constraint.

```
SELECT DISTINCT ?Business_Asset ?Constraint  WHERE {
    ?Business_Asset rdfs:subClassOf CordaSecOnt:BusinessAsset .
    ?Business_Asset rdfs:subClassOf ?Constraint .
    ?Constraint owl:onProperty CordaSecOnt:hasConstraint .
    { ?Constraint owl:someValuesFrom CordaSecOnt:Confidentiality . }
    UNION
    { ?Constraint owl:someValuesFrom CordaSecOnt:Integrity . }
    UNION
    { ?Constraint owl:someValuesFrom CordaSecOnt:Availability . }
}
```

Threats mitigated: The following query bring the threats that are mitigated by using CorDapp. The query result shows the threats mitigated, vulnerabilities and system assets that target by threats.

```
SELECT DISTINCT ?Threats ?Vulnerabilities ?System_Assets  WHERE {
    ?Threats rdfs:subClassOf ?Vulnerabilities .
    ?Threats rdfs:subClassOf ?System_Assets .
    ?Vulnerabilities owl:onProperty CordaSecOnt:exploits .
    ?System_Assets owl:onProperty CordaSecOnt:targets .
    ?Threats rdfs:seeAlso ?Domain .
    FILTER regex(?Domain, "^Mitigated")
}
```

Threats appear: The following query bring the threats that appear after using Corda-platform. The query result shows the threats appeared, vulnerabilities and system assets that target by threats.

```
SELECT DISTINCT ?Threats ?Vulnerabilities ?System_Assets  WHERE {
    ?Threats rdfs:subClassOf ?Vulnerabilities .
    ?Threats rdfs:subClassOf ?System_Assets .
    ?Vulnerabilities owl:onProperty CordaSecOnt:exploits .
    ?System_Assets owl:onProperty CordaSecOnt:targets .
    ?Threats rdfs:seeAlso ?Domain .
    FILTER regex(?Domain, "^Appeared")
}
```

Countermeasures: List of countermeasures that implements security requirements.

```
SELECT DISTINCT ?Countermeasures ?Implements_Security_Requirements  WHERE {
    ?Countermeasures rdfs:subClassOf CordaSecOnt:Countermeasure .
    ?Countermeasures rdfs:subClassOf ?Implements_Security_Requirements .
    ?Implements_Security_Requirements owl:onProperty CordaSecOnt:implements .
}
```

Security requirements: List of security requirements that mitigates security threats.

```
SELECT DISTINCT ?Security_Requirements ?Mitigates_Security_Threats  WHERE {
    ?Security_Requirements rdfs:subClassOf CordaSecOnt:SecurityRequirement .
    ?Security_Requirements rdfs:subClassOf ?Mitigates_Security_Threats .
    ?Mitigates_Security_Threats owl:onProperty CordaSecOnt:mitigates .
}
```

Received November 24, 2020 , accepted November 28, 2020