OntoEDS: Protecting Energy Delivery Systems by Collaboratively Analyzing Security Requirements

Josephine Lamp, Carlos E. Rubio-Medrano, Ziming Zhao and Gail-Joon Ahn
The Center for Cybersecurity and Digital Forensics
Arizona State University
Email: {jalamp, crubiome, zzhao30, gahn}@asu.edu

***Abstract*—Sophisticated attacks on Energy Delivery Systems (EDS) are no longer theoretical, and instead pose a serious threat to American economies. The EDS community has worked collaboratively to develop security requirements to protect EDS against such attacks, but the documents that contain these requirements are often dense, ambiguous and difficult for humans to understand, resulting in highly subjective security implementations that reduce their effectiveness. Therefore, there is need for a methodology that models and visualizes security requirements enabling quick retrieval, understanding and analysis, along with the evaluation and evolution of the implementations of such requirements within EDS. With this in mind, this paper presents a collaborative tool called *OntoEDS* that provides a well-defined representation of security requirements within ontological representations, and a set of ontology exploration techniques to analyze and evaluate the implementations of such requirements against the current attack surface. We also present a case study exemplifying the usefulness of our tool surrounding the series of EDS attacks that occurred in Ukraine in 2015 and 2016.**

I. Introduction

Energy Delivery Systems (EDS) can be broadly defined as the network of processes utilized to manage energy transportation, and include the power grid, gas, and oil industries [1]. Such systems contain a large degree of digital automation in order to effectively control and manage their energy processes efficiently. As they are critical aspects of a state or country's economy, they are strong targets for attackers, due to the potential for great harm and cost to the country upon successful attack. Multiple attacks have occurred with alarming regularity over the past 3 years in Ukraine including the Kyivoblenergo Attack in Kiev (2015) [2], the Prykarpattyaoblenergo Attack in Ivano-Frankivsk region (2015) [2] and the most recent Ukrenergo Transmission Station Attack (2016) [3], and the propensity for attacks such as these to occur in the United States is not unlikely. For example, in the second installment of the January 2017 Quadrennial Energy Review Report on the state of American Energy and EDS, produced by the U.S. Energy Department, the report highlights the danger U.S. grids are under for cyberattacks similar to those seen in Ukraine [4].

Not only are EDS attacks a reality, they are becoming increasingly sophisticated and utilizing more advanced and autonomous tools. As an example, the 2015 Ukraine attack was performed manually utilizing a remote connection, whereas the most recent 2016 attack in Ukraine utilized a highly sophisticated and adaptable piece of malware that learns about an EDS system in order to automatically maximize the amount of damage performed. The sophistication of such malware takes advantage of and targets the EDS on a multitude of levels, from network components to end devices, enabling increased cost to the system. As a result, in order to deter and mitigate these attacks there is a strong need to proactively protect the entire EDS system on a variety of levels of abstraction and granularity, especially considering the innate complexity of energy delivery systems.

In order to better prepare EDSs against such attacks, the EDS community has been working collaboratively to define specific security requirements for EDS. Diverse organizations have released a variety of documentation that specify security requirements at different levels of abstraction, and covering different components and pieces of EDS. However, these documents are lengthy, dense, sparse, and highly-subjective, which makes them difficult to digest and understand, often resulting in subjective and varied interpretations of security implementations, ultimately limiting their effectiveness against attacks.

Therefore, there is a need for security requirements to be modeled and visualized in an easy-to-use and understandable manner, that allows for quick retrieval as well as the integration of multiple requirements from different organizations in order to get a whole, cohesive picture of security for the entire system. In this way, different stakeholders within the context of EDS, namely operators, officials and security officers, can understand the security requirements attained to them. Furthermore, it may aid in security requirement evaluation and act as a foundation for the development of additional tools, such that experts within the field can perform intense analysis based on the following criteria: 1) determine if current security requirements have enough coverage to deter and mitigate existing attacks, 2) determine the ability of requirements to be successfully implemented and their effectiveness against attacks, and finally 3) if they are able to adapt and cope with the current threatscape, as well as evolve safely as EDS infrastructures themselves evolve.

In order to meet the afore-mentioned goals, we propose a collaborative EDS-Ontology-Engine (*OntoEDS*) tool that models security requirements in a well-defined, comprehensive and extensible ontological representation, allowing for stakeholders and security experts in the EDS field to model and understand security requirements, their interdependencies and their implementations. The tool also utilizes a set of

ontology exploration techniques, a.k.a. *projections*, allowing for easy, tailored retrieval and synthesis of requirements that the stakeholder can utilize and analyze. Finally, *OntoEDS* provides a solid foundation for collaboratively developing and categorizing security implementations based on the requirements that they address; for example, it may provide guidelines for effectively implementing firewalls in the context of EDS networks.

In this paper, we provide the following contributions: 1) we provide a methodology for analyzing and synthesizing security requirements contained within large, dense documents and modeling them into an ontology, 2) we provide a well-defined representation of a set of security requirements from 7 key EDS documents, and 3) we provide the methodology for effectively retrieving, utilizing and analyzing such requirements through the use of our ontology exploration techniques. Finally, 4) we provide a sample case study displaying how our proposed tool can be used to collaboratively adapt and defend against serious emerging threats such as CrashOverride [3], a sophisticated piece of malware used in the Ukraine 2016 attack.

This paper is organized as follows: we start by briefly reviewing some important background topics, along with a running example and some other key considerations for our approach in Section II. We elaborate on the problem statement in Section III and our approach and case study is described in Section IV, followed by some related work behind the inspirations for our approach in Section V. In Section VI, we conclude the paper and elucidate the future direction of our work.

II. Background

Ontological Representations. Ontologies are utilized to conveniently model complex real-life domains and scenarios in a structured, intelligent manner, allowing for both computers and humans to understand their concepts [5]. They leverage the basic notions of entities and properties, in which entities represent the objects of a domain and properties define the relationships amongst such entities. These representations are ideal due to their ability to cohesively combine and integrate information from diverse sources, i.e., complex documentation from various unique organizations. They provide a flexible knowledge structure that can accurately represent a variety of objects and data types, and can easily be extended upon the need for inclusion of more material. The overall amount of time it takes to design, implement and use the ontology is relatively minimal, making them ideal for research purposes.

One of the leading semantic languages used to model ontologies is the Web Ontology Language (OWL) [6], a flexible and expressive schema developed for ontologies that offers a range of capabilities to model and describe data and its relationships. In addition, the Simple Protocol and RDF Query Language (SPARQL) [7] is a query language that relies on the concept of *links* to navigate through the entities and properties contained within an ontology, and works well with OWL ontologies. SPARQL links prove useful as they utilize the predefined properties and their associations between entities to combine and compare them such that new and interesting relationships, not originally specified within the ontology, are found.

Sophisticated EDS Attacks. As mentioned in Section I, EDS attacks are no longer a theoretical issue, as recent attacks including the Ukraine ones in 2015 and 2016 clearly indicate. Such attacks are becoming more sophisticated as illustrated by the transition from manual remote manipulation of the system utilized in the 2015 Ukraine attack, to the utilization of a sophisticated, automated malware tool in the 2016 attack. In the latest attack, the malware utilized was called CrashOverride, a specialized framework specifically developed to target electric grids [3]. CrashOverride is flexible and extensible, such that it is not tailored for any specific grid configurations (and instead is compatible with a variety of grid protocols.) It is a modular tool that includes backdoor, launcher, and payload modules, used to access and infect the system, load individual modules into the system, and perform specific functions (such as wipe data, take over the controls of an end EDS device and masquerade as the host or manipulate other EDS components within the system,) respectively. What makes CrashOverride interesting is the fact that the malware (through use of its payload modules) is able to understand, communicate with and even control the actual protocols used by end energy delivery devices such as Programmable Logic Controllers (PLCs) [8] and Remote Control Units (RTUs) [9]. Even more worrisome, it automatically deploys and begins to take over the system, without remote control or initiation from a human attacker, and could be deployed and used in multiple sites simultaneously. With such capabilities, this malware poses a serious threat and has the potential to incite a large degree of damage to an EDS.

III. Problem Statement

As mentioned in Section I, the EDS community has released a variety of security documentation for EDS infrastructures, in an effort to deter targeted attacks. These documents are large and dense, making them difficult to analyze and understand. They contain a great variance in the specific range of content and EDS components they elucidate, and they do not integrate together to create a comprehensive picture of the suggested state of security for EDS. As such, a siloed, fragmented set of security requirements may be implemented, resulting in a less cohesive (and thereby effective) overall security system than one had been implemented comprehensively.

In addition, although these documents are consistently updated and reviewed, the requirements may become out of date or change in relevance and/or priority due to the rapidly changing threat landscape (threatscape); these documents cannot adaptively change to meet the needs of the latest attack types, threatening EDS. Finally, there is a plethora of knowledge and experience contained within experts and EDS stakeholders that may not be properly captured within the security requirements and documentation. For example, whereas documentation may simply list or describe security requirements, experts may have additional insights on specific scenarios or settings that warrant

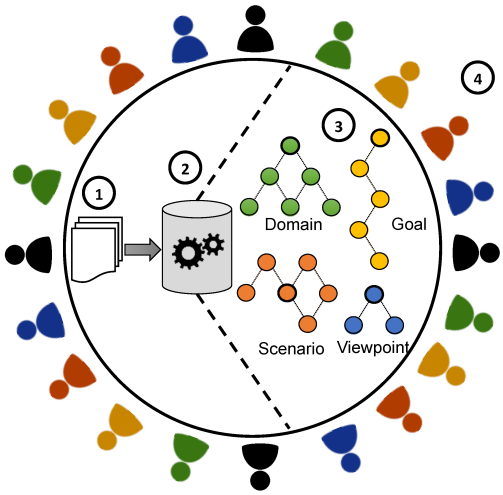

Fig. 1. A depiction of *OntoEDS*, a collaborative requirements-based tool utilized to model, evaluate and analyze security requirements for EDS stakeholders. Reputable security requirements (1) are intelligently modeled in an ontological representation (2) and projections, including goal, domain, scenario and viewpoint traversals (3), are utilized to aide EDS stakeholders in requirements understanding and analysis, allowing for collaborative decision making in terms of requirement priority and EDS infrastructures (4).

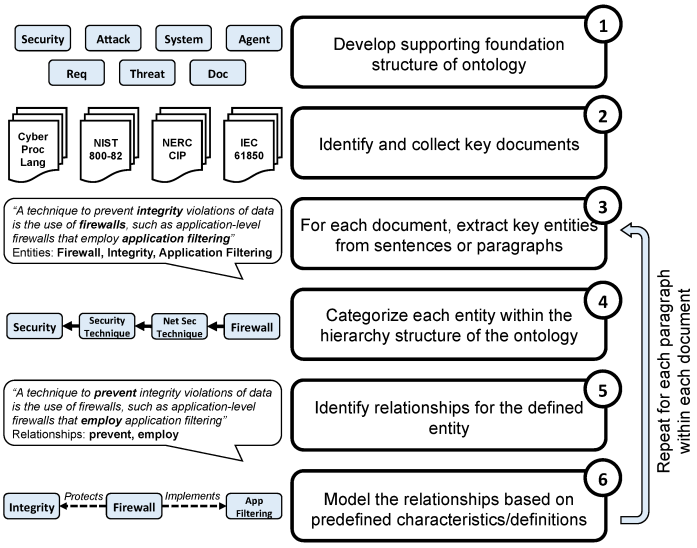

Fig. 2. The process of document modeling within the ontology. The base structure of the ontology entities and high level relationships is developed (1), and the key documents to be modeled are identified (2). Next, in (3) through (6), for each document, entities are extracted and categorized within the hierarchy, and relationships for each entity are identified and modeled.

the use of various security measures over other ones based on prior knowledge related to current threats, or organizational constraints such as time, resources or financial constraints.

IV. Our Approach: Collaborative Security Requirement Modeling and Analysis

As introduced in Section III, with the number and sophistication of attacks on EDS increasing, there is a strong need for the modeling, retrieval and analysis of security requirements such that EDS operators and stakeholders can more easily understand and evaluate such requirements in terms of their coverage, implementations and effectiveness, along with their ability to evolve and adapt to protect against new threats and attack types, in addition to adapting safely with how EDS infrastructures themselves may change. In order to meet these goals, we present *OntoEDS*, a collaborative requirements-based tool utilized to effectively and comprehensively model security documents from diverse sources, as well as retrieve and evaluate such requirements using our ontology exploration techniques. A depiction of *OntoEDS* is given in Figure 1: EDS documents (1) from diverse, reputable sources are modeled in our ontology (2) using automated and manual ontological engineering processes, explained in greater detail in Section IV-A. Next, in (3), ontological exploration techniques, a.k.a. *projections*, are utilized to aide EDS stakeholders in analysis and understanding of security requirements. Such projections include Domain Traversals, Goal Traversals, Scenario Traversals and Viewpoint Traversals, and are explained in greater detail in Section IV-B. Finally, using the *OntoEDS* tool as a foundation, EDS stakeholders in (4) are able to develop tools to examine, analyze and evaluate security requirements and their implementations within EDS infrastructures in order to as a community make decisions about new best practices, prioritize security requirements, and effectively adapt to new attack types and threats. For consistency, a running example surrounding the security concept of a *Network Firewall* will be utilized throughout this section.

A. Ontology Development and Methodology

Document Modeling. In order to comprehensively extract and model the key requirements within the ontology, both manual and automated processes were utilized, with some inspiration drawn from the ontological engineering methodologies described in [5]. Figure 2 gives an overview of this process. To begin, an overall structure to form the base of the ontology was constructed utilizing expert opinion, and generic concepts that would define the overall relationships between concepts within the ontology as shown in Fig 2 (1). For example, within our ontology the 7 key entities of *Security*, *Requirement*, *Attack*, *Threat*, *System Component*, *Documentation* and *Agent* were first modeled, with the defining relationships between each of these concepts defined and also modeled within the ontology. For example, the concept of *Threat* has the relationship isRealizedAs the entity *Attack*, and the entity *Security* has the relationship counteracts the concept *Attack*. Once the base structure of the ontology is created, a set of documents comprising the core of best practices, security implementations and requirements originating from reputable EDS organizations was gathered and evaluated as in Fig. 2 (2). Each document is analyzed page by page, paragraph by paragraph, to extract the key information necessary to model within the ontology. For each paragraph or sentence, entities are identified [Fig. 2 (3)]; these are often nouns within a sentence. For example, in a sentence from the NIST 800-82 standard [9] stating *"A technique to prevent integrity violations of data is the use of firewalls, such as*

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?root
WHERE { ?root rdfs:subClassOf owl:Thing }
```

Listing 1. SPARQL Query for All Root Nodes

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX eds: <http://www.semanticweb.org/ontologies/OntoEDSOntology#>
SELECT ?sub
WHERE { ?sub rdfs:subClassOf+ eds:Requirement }
```

Listing 2. SPARQL Query for All Subclasses of a Node

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX eds: <http://www.semanticweb.org/ontologies/OntoEDSOntology#>
SELECT DISTINCT ?prop ?range
WHERE { ?prop rdfs:domain eds:concept ;
rdfs:range ?range }

SELECT DISTINCT ?prop ?dom
WHERE { ?prop rdfs:range eds:concept ;
rdfs:domain ?dom }

SELECT DISTINCT ?prop
WHERE { ?prop rdfs:subClassOf eds:concept }
ORDER BY (?prop)
```

Listing 3. SPARQL Query for All Domain/Range/Subclass Relationships of a Node

application-level firewalls that employ application filtering," the entity *Firewall* would be extracted. Next, in step (4) as shown in Fig. 2, the entity is classified and categorized within the overall hierarchy of the ontology, based on the previously defined base structure. For example, within our ontology, the concept would be broadly categorized under one of the 7 base categories; *Firewall* would be categorized under the key category *Security*. From there, using a standardized ontological engineering process, the concept would be placed in the exact hierarchy based on its specialization or *superclass* of surrounding entities already modeled within the ontology. In the case of the entity *Firewall*, it would be categorized underneath the class *Security Techniques*, more specifically as the direct subclass of *Network Security Techniques*. Next, the relationships that are connected to that entity are identified [Fig. 2 (5)]. These are often verbs within a sentence, i.e. within the previously-mentioned sentence, the relationships prevent and employ would be identified. From there, such relationships can be modeled within the ontology based on a standardized process [Fig. 2 (6)]. In the case of the *Firewall* entity, the relationships *Firewall* protects *Integrity* and *Firewall* implements *Application Filtering* would be added. This process is repeated until the entire document is modeled.

The above comprehensive process may be lengthy in time spent to create an ontology model, and as such automated processing using Natural Language Processing [10] was also conducted to speed up this process. Once a basis structure of the ontology is created (for example, the 7 main categories of entities described in greater detail in the succeeding section), sentences can be analyzed for their entities and relationships and easily fit into the rest of the ontological structure.

Current State of our Ontology. Our current *proof-of-concept* ontology effectively models the security requirements contained in 7 major EDS security documents from diverse organizations including the Cybersecurity Procurement Language for Energy Delivery Systems [8] developed by the Energy Sector Control Systems Working Group (ESCSWG), the NIST 800-82 Special Publication [9] and the North American Electric Reliability Corportation Critical Infrastructure Protection (NERC CIP) standards [11]. The ontology comprises more than 300 pages total and includes 600 entities with over 1,700 relationships modeled between them. There are a total of 7 core categories of concepts that all entities are subclasses of, including Requirements, Security, Attacks, Threats, Documentation, System (components) and Agents (a.k.a. system actors). For the manual modeling of our ontology, Protege, an easy-to-use ontology development tool and interface developed by Stanford in 1999 was used to manually model the ontology [12]. In addition for the automated processing of document text, GATE was utilized [10].

B. Ontology Exploration Techniques

Ontology Engine Overview. As described in Section I, once the documents and security requirements are intelligently modeled within the ontology, a way to easily understand and analyze such requirements is needed. This is accomplished through the use of our ontology exploration techniques a.k.a. *projections*, deployed within the *OntoEDS* ontology engine as shown in Fig. 1 (3). The ontology engine was written in Java and utilizes SPARQL queries to traverse the actual OWL ontology, conducted through the Apache Jena API [13]. The specific SPARQL queries utilized throughout the engine are shown in Listing 1, 2 and 3. Listing 1 displays the query utilized to obtain all top-level root nodes of the entire ontology (in this case the 7 key categories mentioned previously), Listing 2 shows the query to obtain all subclasses of a starting node, and Listing 3 pulls all domain, range and subclass relationships from a starting node. Each of these queries was utilized within the engine, and will be explained in greater detail in the next sections.

For the purpose of our ontology engine, four projections were developed: Goal Traversals, Domain Traversals, Scenario Traversals and Viewpoint Traversals. To explain the functionality and use of such techniques, we will be using a continuation of our running example surrounding the concept of *Firewalls*. In order for our Goal Traversals to work properly, a metadata tagging preprocessing step was first conducted in which each entity in the ontology was tagged based on its core superclass (one of the 7 base categorizations, such as Security, Requirement or Attack). The purpose and necessity of such process is explained later on in the Goal Traversal methodology described below. As show in Algorithm 1, the ontology is first loaded into the engine in lines 1 and 2. Next, a SPARQL query (Listing 1) is used to retrieve all root nodes of the ontology as shown in line 3, and for each root, all subclasses are identified by using another SPARQL query

Algorithm 1 Preprocess and Add Metadata Tags

1: $m \leftarrow$ getOntologyModel()
2: loadData(m, $true$)
3: $roots \leftarrow$ getQuery($rootNodes$))
4: **for all** $r_i \in roots$ **do**
5: $subs \leftarrow$ getQuery($subclassNodes(r_i)$)
6: **for all** $s_i \in subs$ **do**
7: $s_i \leftarrow$ add(createAnnotationProperty($r_i.name$))
8: **end for**
9: **end for**
10: exportOntology(m)

Algorithm 2 Goal Projection

1: read($concept$, $category$, $complexCategory$, $iterations$, $complexIterations$)
2: **for all** $i_i \in iterations$ **do**
3: **if** $complexCategory$ && $complexIteration$ **then**
4: $category$ = $newCategory$
5: **end if**
6: $nodes \leftarrow$ getQuery(domainRangeSubclassNodes($concept$))
7: $finalNodes \leftarrow$ filter($nodes$, $category$)
8: **end for**
9: **return** $finalNodes$

(Listing 2) in line 5. Each subclass is given a metadata tag based on the root node type (denoted through the creation of an AnnotationProperty in Apache Jena) as in line 7. Finally, the ontology changes are saved and exported back into an OWL format in line 10.

Goal Projections. Within EDS systems, stakeholders are often concerned with the achievement or understanding of a specific goal, and as a result the core of our ontology engine relies on goal traversals. Within the context of our approach, goals are defined as objectives the system should achieve, with the ultimate goal to achieve a state of security. For example, types of goals could include securing or protecting system components, system adherence to requirements, implementing security techniques or features, defending against an attack type, determining agent responsibilities, identifying purposes or properties of system components and protecting security principles. Goals are formulated based on user expectations or needs, and our *OntoEDS* tool utilizes such Goal Projections to effectively answer the user's queries.

Upon initial development of Goal Projections, it was necessary to determine how to translate human conceptualized goals into something the ontology engine could understand. The engine itself relies on starting with a specific entity to begin a traversal, and as such the human goal was needed to be translated into a machine understandable starting node. In order to do this, the tool utilizes the metadata tags created in the pre-processing step. User queries are translated into a <*concept* + *category*> tuple, in which the *concept* elucidates the starting node for the traversal and the *category* part describes the end result group the query is expected to return, correlating to a specific metadata tag.

The Goal Projection routine is described in Algorithm 2. As shown in line 1, the engine first expects a concept (starting node), category (metadata tag), and number of iterations for the traversal to run. Iterations specify how many links (or SPARQL relationships, including subclass or domain/range links) away from the starting node the user would like to traverse. For example, if the user specifies an iteration number of 3, the engine will traverse until it reaches nodes that are more than 3 links away from the starting concept. There is also the option to include a complex category and the number of iterations the secondary category will be used. If the user specifies they want to *Protect Integrity* (concept = *Integrity*, category = *Security*) with the secondary goal of determining what security implementations found may apply to what system components within the system (concept = Security, category = System Components) at iteration 3, then the engine will traverse the first two iterations surrounding the goal of security for integrity, and on the third iteration traverse with the goal of finding system components for the security measures found in the previous traversal. Within Algorithm 2 line 2, the traversal runs for the specified number of iterations, and performs a check in lines 3-5 for the complex category. If the complex category iteration has come up, then it sets the new category choice. Next, in line 6 a SPARQL query (Listing 3) is performed to find all of the domain, range and subclass relationships connected to the starting concept node. From there in line 7, the found nodes are filtered based on their metadata tag. If it relates to the specified category, the node is added to the traversal. Finally, after all of the iterations are completed, the projection returns the resulting set of nodes and their paths found in the traversal.

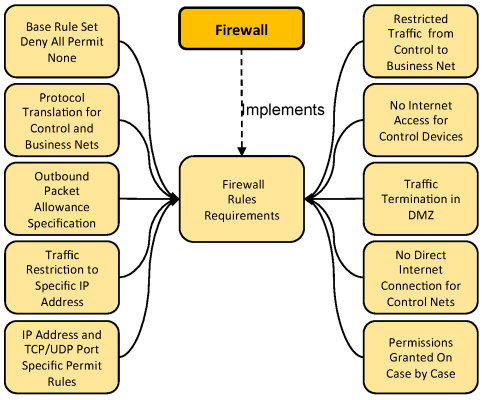

Fig. 3. A simple Goal Projection for the concept of *Firewall* and the category of *Requirements*. The traversal identifies specific requirements related to Firewall Rules, such as the a requirement that specifies the *Base Rule Set Deny All Permit None*.

The Goal Projection methodology traverses from the starting concept and uses filtering to make choices on path traversal using the categorization specification in order to get to the final node and traversal. The user can conduct simple goal traversals, in which the user specifies the starting node and a single category. As displayed in Fig. 3, a simple goal traversal was completed using the concept of *Firewall* and the category

Algorithm 3 Domain Projection

1: read($concept$, $iterations$)
2: **for all** $i_i \in iterations$ **do**
3: $nodes \leftarrow$ getQuery($subclassNodes(concept)$)
4: **end for**
5: **return** $nodes$

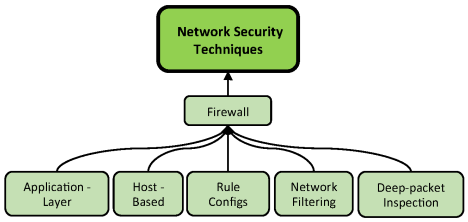

Fig. 4. A Domain Projection surrounding the entity *Firewall*. Solid lines are used to show inheritance class relationships, i.e. *Firewall* is a subclass of *Network Security Techniques*, and *Rule Configs* are a subclass of *Firewall*.

of *Requirements*. In addition, the user can conduct complex goal traversals, in which the user can specify multi-part goals by specifying multiple categories with the concept they would like the engine to filter by. For example, in the case of the Firewall, the user may first specify they want to conduct traversals around the category of *Requirements* and then they would like to learn what specific system components such requirements are applicable to, and specify a second category of *System*.

Domain Projections. Domain Projections are utilized to understand the subclass relationships related to a node. In the context of our approach, a domain is defined as the immediate hierarchical tree surrounding a specific node. The Domain Projection algorithm works by beginning with a starting node, and traversing from the starting node down through subclasses of the node (top-down approach) as many iterations as the user would like, or until all leaf nodes are found. An example surrounding the concept *Firewall* is shown in Fig. 4. As shown in, Algorithm 3, in line 1 the user supplies the specific starting concept as well as the number of iterations they would like the traversal to run. Then, in lines 2-5, for each iteration a SPARQL query (Listing 2) is used to obtain all of the subclass relationships of that node (or set of nodes), and when the iterations are completed, the final nodes and tree are returned.

Scenario Projections. Within the context of our approach, scenarios are defined as facts describing a system that may include agent behavior and environmental context to be used for discovery or validation of system requirements. The scenario may identify dependencies between the system and its environment, and provides a storyline of events describing system operation in relation to a specific entity. Scenario Projections give a broad understanding of the immediate area of relevance to a concept, and serve as an introductory knowledge exploration of a node or set of nodes. It is the intention that more specific traversals should be completed using Goal or Viewpoint Traversals. It is important to note here that due to the larger number of related nodes pulled surrounding a

Algorithm 4 Scenario Projection

1: read($concept$, $iterations$)
2: **for all** $i_i \in iterations$ **do**
3: $nodes \leftarrow$ getQuery($domainRangeNodes(concept)$)
4: **end for**
5: **return** $nodes$

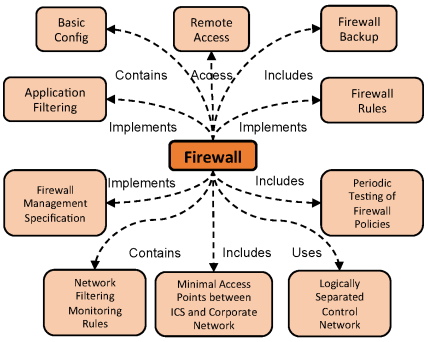

Fig. 5. A Scenario Projection displaying the surrounding nodes of the concept *Firewall*. Dotted lines are used to represent domain and range relationships between entities. For example, the *Firewall* uses the *Logically Separated Control Network*, contains a *Basic Config* and implements *Firewall Rules*.

starting concept, for Scenario Projections it often does not make sense to complete more than one (or sometimes two) iterations, as the resulting traversal becomes too large, with the specificity of information too broad to prove useful to the user. This was determined based on experimentation with the traversal type. Scenario Projections traverse both top-down (up the tree into superclasses) and bottom-up (down the tree into subclasses) from a node, and pull all relationships immediately surrounding that node, of which an example is shown in Fig. 5. As shown in Algorithm 4, in line 1 the user enters in the concept and number of iterations wished to complete. Next, in lines 2-5, for each iteration all of the domain, range, sub and super class relationships surrounding that node are pulled from the ontology using a SPARQL query (Listing 3), and the final set of nodes and path traversals are returned.

Viewpoint Projections. A viewpoint is defined as the status or mental position one takes when using a system, often related to that user's role within the EDS system, in our approach. Viewpoint Projections are utilized to understand the specific frame of reference related to an individual agent, often in terms of specific responsibilities they may undertake. For the traversal within the engine, Viewpoint Projections are a specialized form of Goal Traversal, in which the concept is the specific agent and the category that gets filtered by is related to specific agent functions, such as *agentResponsibilities*. An example relating to the concept of *Firewall* are shown in Fig. 6. The algorithm, shown in Alg. 5, looks very similar to the Goal Projection (as in Alg. 2): In line 1, the agent is specified, and in line 2 a SPARQL query (Listing 3) to identify all nodes related to that agent is found. Finally, in lines 3 and 4, the nodes are filtered based on the category *agentResponsibilities*, and the final node set is returned.

Algorithm 5 Viewpoint Projection

1: read($agent$)
2: $nodes \leftarrow$ getQuery(domainRangeSubclassNodes($agent$))
3: $finalNodes \leftarrow$ filter($nodes$, $agentResponsibilities$)
4: **return** $finalNodes$

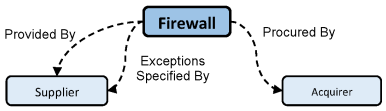

Fig. 6. A Viewpoint Projection showing the related agent responsibilities for the entity *Firewall*. For example, the *Firewall* is providedBy the *Supplier* and procuredBy the *Acquirer*.

C. Projection Traversal Use.

Targeted Knowledge Acquisition. As mentioned in Section I, our tool *OntoEDS* can be utilized to allow EDS operators to gain a better understanding of security requirements and their interdependencies, along with the ability to evaluate and analyze specific requirements and concepts. These user goals can concretely be obtained through the use of our projections. In terms of targeted knowledge acquisition, Goal Traversals allow the straightforward fulfillment of user queries that may be related to requirements, standards, threats and/or system components. A user can formulate a goal based around specific questions they would like to answer, and by utilizing our Goal Traversal, a query can be returned that effectively supplies information to satisfy the user's initial goal or question. For example, as depicted in Fig. 3, a user may wish to identify and understand the specific requirements contained in the documentation related to the asset *Firewall*. In addition, as a more specialized form of Goal Traversals, users can take advantage of the Viewpoint Projections to determine specific agent responsibilities or immediately relevant aspects to a particular user viewpoint, i.e. as displayed in Fig. 6.

Knowledge Exploration. Our projections also allow for the more ambiguous retrieval of information surrounding knowledge exploration. Scenario Traversals allow for the introductory understanding of the immediate frame of reference and relevant concepts surrounding an entity. It enables the user to gain a quick understanding of the broad picture of that concept and its surrounding relationships. As shown in Fig. 5, a quick overarching view of the entities and relationships immediately surrounding the entity of *Firewall* is displayed. The user can quickly see types of concepts that access the Firewall (*Remote Access* and *Network Access*), elements contained within the Firewall (*Basic Config* and *Network Filtering and Monitoring Rules*, the types of entities the Firewall may implement (*Firewall Management Specification* and *Firewall Rules*), among many other relationships and entities. Furthermore, the Domain Projections allow the user to understand the domain and hierarchy of subclasses related to that concept, to gain an understanding about concepts and their inheritance patterns or characteristics in relation to the categorization of an entity. For example, as shown in Fig. 4, one can quickly and easily see the super- and sub- classes related to the *Firewall* concept, including *Network Security Techniques* for the prior and types of Firewalls such as *Application-Layer* and *Deep-packet Inspection* for the later.

Requirement Analysis. Finally, projections allow the user to perform requirement and concept analysis as the user can understand concepts from different perspectives and angles (for example understanding an entity in terms of its hierarchy with a Domain Traversal vs understanding an entity in terms of its surrounding relationships with a Scenario Traversal). An entity can be analyzed based on specific goals to get a better understanding of element of an entity and how they may relate to an end goal such as security or threats, and a plethora of information surrounding specific agents and their responsibilities can also be identified utilizing Viewpoint Projections. Moreover, projections allow EDS operators to gain a broader understanding of the interdependencies and inter-relationships between and amongst entities, in order to gain a more comprehensive understanding of the entire EDS system, and how that may relate to security requirements. As shown in the proceeding projection examples surrounding the *Firewall* entity, various traversals can help to illuminate different aspects and perspectives of the entity and its relations. In summary, an example of how the user may use each projection is as follows: Starting with the Scenario Traversal, the user can gain a quick high-level understanding of the immediate concepts relevant to the *Firewall* entity; these include information about implementation assets, subcomponents and concepts that utilize the entity, among others. From the Goal Traversal, one can identify specific requirements applicable to the *Firewall* asset, including the requirements of *Traffic Termination in DMZ*, *Base Rule Set Deny All Permit None* and *Permissions Granted on Case by Case Basis*. Continuing on, from the Domain Traversals, the user can identify more detailed specifications about the other types of Firewalls there are, such as *Host-based* and *Network Filtering*, and can utilize the Viewpoint Traversals to gain a quick understanding of the agent responsibilities surrounding the *Firewall* asset, in this case that the asset is provided by the *Supplier* and procured by the *Acquirer*.

D. Case Study

In order to elucidate the usefulness and applicability of *OntoEDS* to EDS, we illustrate a case study surrounding the series of attacks that occurred in Ukraine in 2015 [2] and 2016 [3], and present the identification of missing requirements, found utilizing our tool.

2015 Ukraine Attack. After the attack involving remote manipulation in 2015 occurs, a group of EDS stakeholders including operators and security officials gather together to analyze the current set of requirements contained within the literature. They want to evaluate the coverage and effectiveness of such requirements based on the current threatscape and information learned from the latest attack, and utilize our *OntoEDS* tool to complete this task. They decide to focus on two key areas surrounding network security that were integral

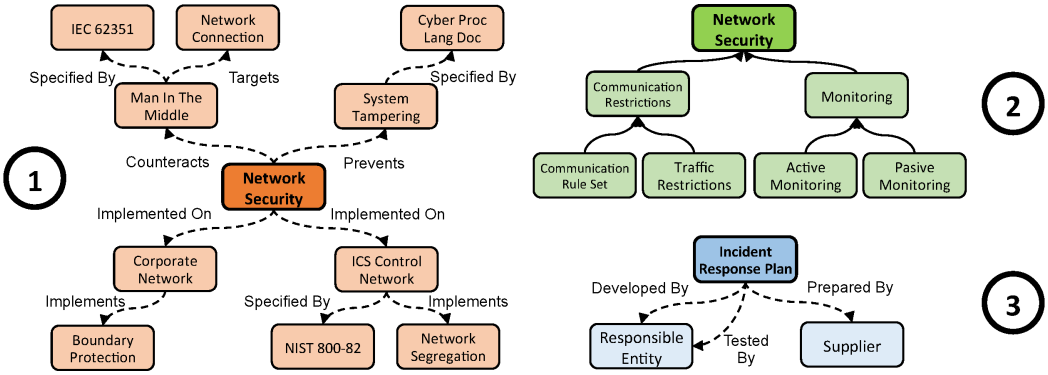

Fig. 7. A Scenario (1), Domain (2), and Viewpoint (3) Projection performed by EDS collaborators after the 2015 and 2016 Ukraine attacks.

in allowing the successful completion of the 2015 attack: 1) the lack of two-factor authentication between the business network and a VPN allowing access to the industrial control network, and 2) the lack of network intrusion detection efforts within the industrial control network that failed to identify network abnormalities.

First, they begin by exploring what network security requirements are contained within the documentation. They utilize *OntoEDS* to perform a Scenario Projection surrounding the concept of *NetworkSecurity*. After two iterations using our tool, they see a wide picture of the surrounding concepts, a subset of which is shown in Fig. 7 (1). The traversal includes the types of attacks *NetworkSecurity* may counteract such as *ManInTheMiddle* and *UnauthorizedAccess*, threat types such as *SystemTampering* that it may prevent, system components it relates to such as *CorporateNetwork* and *ICSControlNetwork*, and other security measures such as *NIDS* and *Network Connection Authentication*. Moreover, the second iteration supplies additional information surrounding these nodes, as shown by the second level of concepts in Fig. 7 (1). These include the types of documentation that specify such entities, including *IEC62351* and *NIST800-82*, and additional, more-specific security measures such as *BoundaryProtection* and *NetworkSegregation*, among other concepts and relationships. In this way, the collaborators gather a broad view of the types of information and relationships that are connected to this concept and what these requirements may specify in terms of such concepts.

Next, they want to gain a better idea of the specifications and sub-types of *NetworkSecurity* contained within the requirements, in order to understand more in-depth what specific entities devolve directly from the concept. By performing a Domain Projection, they are able to identify subclasses about specific types of Network Security including *Communication-Restrictions*, *NetworkProtocol* specifications, *NetworkMonitoring*, and *NetworkSecurityZones*. A subset of the projection is shown in Fig. 7 (2). By looking at the following subclasses, they are able to see the depth of detail the requirements specify related to these concepts (and Network Security as a whole). For example, under *NetworkMonitoring*, there is a variety of subclasses related to they types of monitoring, including *ActiveMonitoring* and *PassiveMonitoring*, along with the concept *NIDS*, types of NIDS, NIDS architecture specifications, NIDS sensors, and NIDS configurations originating from various documentation sources including the NIST 800-82 standard [9], and the Cybersecurity Procurement Language for EDS document [8].

After evaluating each of these concepts, they determine there is good coverage, breadth and depth of the topics surrounding *Network Security*; based on the use of our tool's projections, the collaborators identified a wide range of *Network Security* specifications, requirements and security measures, (beyond that of just NIDS,) that may be implemented to better protect EDSs. From this, the collaborators use the knowledge they have learned and the specific requirements retrieved to make group security recommendations for EDS infrastructures, in order to protect against the specific attack vectors utilized in the 2015 attack.

2016 Ukraine Attack. A year passes, and now the EDS collaborators reconvene to analyze the 2016 Ukraine attack involving the use of CrashOverride, a sophisticated piece of malware described in greater detail in Section II. They wish to evaluate the set of requirements from the literature in the context of this new threatscape and based on these new attack vectors, using our tool as a proxy to complete this process. Upon discussion, they decide that a key security feature especially relevant to the recent attack is the use of Incident Response Plans. As cited in [3], one of the faults of the energy grid systems was the lack of clear plans and understanding of ways to handle and mitigate the attack consequences. Wishing to implement this measure immediately, the collaborators want to identify what agents are specifically responsible for the development, execution and maintenance of such plans. In order to do this, they conduct a Viewpoint Projection through *OntoEDS* for *IncidentResponsePlan* and receive the traversal shown in Fig. 7 (3). As a result, they quickly identify that the *ResponsibleEntity* as defined in the NERC CIP standards [11] is responsible for the development and testing of the plan, and the plan is supplied and utilized by the *Supplier*. In this way they are able to efficiently begin the implementation of such plans in various EDSs by working with the appropriate people.

The collaborators continue to analyze the malware and the

vulnerabilities used for the attack's successful completion. Based on such analysis, they identify the weakest link utilized by CrashOverride was the manipulation of EDS protocols, including IEC 104, IEC 61850 and the OPC standards. In this instance, the collaborators decide to focus on protocol and end-device security as opposed to network security as emphasized in the year prior. In order to identify what types of security measures may be contained within the requirements related to such protocols, they use *OntoEDS* to perform multiple goal traversals for each protocol. For example, to determine security requirements of the protocol IEC 104, they conduct a goal traversal using the concept of *IEC 104* and the category of *Security*. However, no entities are returned for any of the protocol-related traversals, indicating potential missing requirements that need to be addressed.

Identifying Missing Requirements. Upon conducting such goal traversals as the ones elucidated above, we have identified that no requirements or security specifications are described related to the protection of EDS protocols within current best practice documentation. Moreover, upon additional research, the protocol specification documents themselves do not specify security measures either. For instance, the IEC 104 document states that "security mechanisms are outside the scope of this document" [14]. These are missing requirements that should be addressed; there is a definite need for clear security measures of EDS protocols as these protocols contain security vulnerabilities that need to be mediated, especially considering the success of the CrashOverride attack. There is a plethora of research and academic suggestions for the security of these protocols that has been published over the last couple of years. As such, this research needs to be synthesized into a standardized, best-practice document that can be deployed and utilized to provide recommendations for, and requirements of, the security of protocols for EDS systems. Additionally, such protocols and their measures can be utilized in the creation of new requirements, as they may be indicative of potential attacks or threats occurring. For example, according to [3], when the OPC protocol is manipulated it will look abnormal and generate increases in network traffic. As such, this information could be turned into a security requirement and/or traffic monitoring rule contained within NIDS or other network monitoring tool that checks for increases in traffic related to such a protocol.

As illustrated above, *OntoEDS* can allow users to adaptively change their focuses of analysis based on new attack types or changes in the threatscape. They can proactively and easily analyze different requirements at various levels of abstraction along with their interdependencies over time, through the use of our ontology and ontology exploration techniques. Furthermore, as shown by our above findings surrounding missing requirements for EDS protocol security, our tool allows for the analysis, evaluation and identification of requirements, including missing or lacking ones. Ultimately, *OntoEDS* allows stakeholders to evaluate security requirements so they can evolve, adapt and improve just as EDS infrastructures and the threatscape evolves to aide in the protection of such systems.

V. Related Work

Ontological Representations. Ontologies have been widely used in literature to create standardized, well-structured models of various domains, and EDS is no exception. A variety of general ontologies utilize to model security concepts in the context of EDS has been developed: Hieb et al. developed an ontology to model process control systems and their components and processes, utilized in a fault diagnosis algorithm that can indicate areas of fault and potential sources for cyber intrusion incidents [15]. Although they do use some ontological principles to back their ontology modeling, their approach is not requirement-based, and instead uses an intrusion detection-style approach, in which they model and understand the current process and then identify when anomalies occur that may be indicative of a fault. Tebbe et al. developed an ontology of general concepts for security in Industrial Control Systems (ICS) along with specific knowledge principles necessary to know in order to perform security assessments of the system [16]. Each of these approaches is very high-level, utilized to accomplish a specific end-user task. In the case of *OntoEDS*, although the high-level structure of our ontology is similar to these methodologies, we provide a more in depth modeling of specific concepts and their relationships. Furthermore, with the use of our ontology engine and the breadth of information contained within our ontology, we are able to meet a variety of EDS stakeholder uses and goals, beyond that of simply understanding the components of a security assessment.

Threat/Attack Taxonomies. Moreover, there is a variety of work on developing taxonomies for threats or attack types specific to EDS systems. For example, Amin et al. [17] and Bompard et al. [18] developed threat taxonomies, the prior to classify failure types and deficiencies for SCADA systems in order to calculate risks to the system through the use of a game-theoretic framework and the later to categorize and understand threat factors, impacts and changes over time. Fleury et al. [19] developed a taxonomy of attacks, vulnerabilities and damages to ICS based on prior literature surrounding identified threats, attack types and vulnerabilities of ICS. This body of work allows for better understanding of the threatscape and attack vectors surrounding and specific to EDS, but each taxonomy can only be used singularly within its one concept area, and is not based on any standardized documentation or best practices. This methodology makes it difficult to assess interdependencies amongst concepts, i.e., between security mechanisms, EDS system assets and threat or attack types, and does not provide a common ground of understanding through which EDS collaborators can discuss and determine best practices for such taxonomy categorizations. Our tool *OntoEDS* goes beyond that of a singular taxonomy, and instead allows the analysis and retrieval of information from standardized documentation of reputable organizations and a variety of concept areas including threats, attacks, security, requirements and agents. Due to its more comprehensive nature, we are able to identify inter-dependencies between concepts, and can aide collaborators in requirement analysis and retrieval through the

use of our projections, in turn allowing for easier discussions and improvements to such requirements and concepts.

Requirements-Based Frameworks. Recently, a variety of frameworks to model and perform risk assessment of EDS systems using specific requirements and standards has been developed in the literature [1]. Liu et al. [20] developed an asset-based risk assessment framework based on the IEC 61850 standard and Fenz et al. [21] developed an ontological representation of the ISO 27002 standard for use in compliance-checks. However, each of these approaches focus on modeling a specific standard and do not provide a holistic, comprehensive view of a variety of diverse documents and their interdependencies. In addition, they do not provide automated tools that can be utilized by EDS stakeholders to aide in the retrieval, analysis and understanding of security requirements in the context of EDS infrastructures.

VI. Conclusions and Future Work

In this paper, we have presented our methodology and tool *OntoEDS* utilized for analyzing and synthesizing security requirements from diverse documentation, creating well-defined representations of such requirements within an ontology, and our ontology exploration techniques that allow the intelligent retrieval and analysis of such requirements. In this way, EDS stakeholders can model and understand security requirements and their interdependencies, as well as utilize the tool for the tailored retrieval, synthesis and analysis of such requirements. In terms of future work, we are working on developing additional ontology projections to aide in risk analysis and risk quantification, i.e., through the creation of automated risk questionnaires and a risk-scoring methodology based on ontology traversals surrounding an EDS asset, such as a PLC. We are also working on the development of a surrounding security assessment and monitoring framework that can automate the validation and checking of security requirement implementation within EDS infrastructures, such as through the use of processing modules to aide in the monitoring and evaluation of network configurations.

Acknowledgment

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000780 and by a grant from the Center for Cybersecurity and Digital Forensics at Arizona State University.

Disclaimer

References

[1] W. Knowles, D. Prince, D. Hutchison, J. F. P. Disso, and K. Jones, "A survey of cyber security management in industrial control systems," *International journal of critical infrastructure protection*, vol. 9, pp. 52–80, 2015.

[2] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," *SANS ICS Report*, 2016.

[3] D. Inc. (2017) Crashoverride analysis of the threat to electric grid operations. [Online]. Available: https://dragos.com/blog/crashoverride/CrashOverride-01.pdf

[4] US Department of Energy, *Quadrennial Energy Review Transforming the Nation's Electricity System: The Second Installment of the QER*, January 2017. [Online]. Available: https://www.energy.gov/sites/prod/files/2017/02/f34/Quadrennial%20Energy%20Review--Second%20Installment%20%28Full%20Report%29.pdf

[5] S. W. Lee and R. A. Gandhi, "Ontology-based active requirements engineering framework," in *Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific*. IEEE, 2005, pp. 8–pp.

[6] "OWL Web Ontology Language Overview," W3C, February 2004. [Online]. Available: {https://www.w3.org/TR/owl-features/}

[7] W3C, "SPARQL Query Language for RDF," January 2008. [Online]. Available: https://www.w3.org/TR/rdf-sparql-query/

[8] Energy Sector Control Systems Working Group (ESCSWG), "Cybersecurity Procurement Language for Energy Delivery Systems," April 2014. [Online]. Available: https://www.energy.gov/oe/downloads/cybersecurity-procurement-language-energy-delivery-april-2014

[9] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, "Nist special publication 800-82 revision 2, guide to industrial control systems (ics) security," *Gaithersburg, MD, USA: National Institute of Standards and Technology*, 2014.

[10] S. Zaidi, M. Laskri, and A. Abdelali, "Arabic collocations extraction using gate," in *Machine and Web Intelligence (ICMWI), 2010 International Conference on*. IEEE, 2010, pp. 473–475.

[11] NERC, "CIP Standards," 2017. [Online]. Available: http://www.nerc.com/pa/Stand/Pages/CIPStandards.aspx

[12] Stanford University, "Protege," 2016. [Online]. Available: http://protege.stanford.edu/

[13] A. Jena, "Apache jena," *jena. apache. org [Online]. Available: http://jena.apache.org*, 2017.

[14] *IEC 60870-5-104*, International Electrotechnical Commission, Std., Rev. 2, 2006. [Online]. Available: https://webstore.iec.ch/preview/info_iec60870-5-104%7Bed2.0%7Den_d.pdf

[15] J. Hieb, J. Graham, and J. Guan, "An ontology for identifying cyber intrusion induced faults in process control systems," *Critical Infrastructure Protection III*, pp. 125–138, 2009.

[16] C. Tebbe, K.-H. Niemann, and A. Fay, "Ontology and life cycle of knowledge for ics security assessments," in *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016*. BCS Learning & Development Ltd., 2016, pp. 1–10.

[17] S. Amin, G. A. Schwartz, and A. Hussain, "In quest of benchmarking security risks to cyber-physical systems," *IEEE Network*, vol. 27, no. 1, pp. 19–24, 2013.

[18] E. Bompard, T. Huang, Y. Wu, and M. Cremenescu, "Classification and trend analysis of threats origins to the security of power systems," *International Journal of Electrical Power & Energy Systems*, vol. 50, pp. 50–64, 2013.

[19] T. Fleury, H. Khurana, and V. Welch, "Towards a taxonomy of attacks against energy control systems," in *International Conference on Critical Infrastructure Protection*. Springer, 2008, pp. 71–85.

[20] N. Liu, J. Zhang, and X. Wu, "Asset analysis of risk assessment for iec 61850-based power control systems—part i: methodology," *IEEE Transactions on Power Delivery*, vol. 26, no. 2, pp. 869–875, 2011.

[21] S. Fenz, S. Plieschnegger, and H. Hobel, "Mapping information security standard iso 27002 to an ontological structure," *Information & Computer Security*, vol. 24, no. 5, pp. 452–473, 2016.