ExSol: Collaboratively Assessing Cybersecurity Risks for Protecting Energy Delivery Systems

JOSEPHINE LAMP, University of Virginia, USA
CARLOS E. RUBIO-MEDRANO, Texas A&M University-Corpus Christi, USA
ZIMING ZHAO, The University at Buffalo, USA
GAIL-JOON AHN, Arizona State University, USA and Samsung Research, Korea

No longer just prophesied about, cyber-attacks to Energy Delivery Systems (EDS) (e.g., the power grid, gas and oil industries) are now very real dangers that result in non-trivial economical losses and inconveniences to modern societies. In such a context, risk analysis has been proposed as a valuable way to identify, analyze, and mitigate potential vulnerabilities, threats, and attack vectors. However, performing risk analysis for EDS is difficult due to their innate structural diversity and interdependencies, along with an always-increasing threatscape. Therefore, there is a need for a methodology to evaluate the current system state, identify vulnerabilities, and qualify risk at multiple granularities in a collaborative manner among different actors in the context of EDS. With this in mind, this paper presents *ExSol*, a collaborative, real-time, risk assessment ecosystem that features an approach for modeling real-life EDS infrastructures, an ontology traversal technique that retrieves well-defined security requirements from well-reputed documents on cyber-protection for EDS infrastructures, as well as a methodology for calculating risk for a single asset and for an entire system. Moreover, we also provide experimental evidence involving a series of attack scenarios in both simulated and real-world EDS environments, which ultimately encourage the adoption of *ExSol* in practice.

CCS Concepts: • **Security and privacy** → **Security requirements**; • **Computer systems organization** → **Embedded and cyber-physical systems**; • **Information systems** → *Collaborative and social computing systems and tools*; **Ontologies**.

Additional Key Words and Phrases: Risk Assessment; Energy Delivery Systems

ACM Reference Format:
Josephine Lamp, Carlos E. Rubio-Medrano, Ziming Zhao, and Gail-Joon Ahn. 2019. *ExSol*: Collaboratively Assessing Cybersecurity Risks for Protecting Energy Delivery Systems. *Digit. Threat. Res. Pract.* 1, 1, Article 1 (January 2019), 24 pages.

1 INTRODUCTION

Energy Delivery Systems (EDS) consist of the network of processes, e.g., software and hardware components, utilized to manage energy transportation, including the power grid, gas, and oil industries [11]. Nowadays, these systems contain a high degree of automation used to efficiently manage the distribution of energy among different geographical regions. Unsurprisingly, since EDS are critical components of a country's economy, they are high caliber targets for cyberattackers.

Authors' addresses: Josephine Lamp, University of Virginia, Charlottesville, Virginia, USA, jl4rj@virginia.edu; Carlos E. Rubio-Medrano, Texas A&M University-Corpus Christi, Corpus Christi, Texas, USA, carlos.rubiomedrano@tamucc.edu; Ziming Zhao, The University at Buffalo, Buffalo, New York, USA, zimingzh@buffalo.edu; Gail-Joon Ahn, Arizona State University, Tempe, Arizona, USA , Samsung Research, Korea, gahn@asu.edu.

2576-5337/2019/1-ART1 $15.00
https://doi.org/

Recently, attacks targeting EDS have become a real danger, such as the multiple attacks that have occurred over the past 4 years in Ukraine, including the Kyivoblenergo Attack in Kiev (2015) [14], the Prykarpattyaoblenergo Attack in the Ivano-Frankivsk region (2015) [14], and the most recent Ukrenergo Transmission Station Attack (2016) [4]. This danger will only continue to grow, and the propensity for attacks to occur in other countries is not unlikely. For instance, in the United States, the U.S. Department of Energy's second installment of the January 2017 Quadrennial Energy Review Report on the State of American Energy and EDS highlights the danger U.S. power grids are under for attacks similar to those seen in Ukraine [22].

In such a context, risk analysis has been proposed as a valuable way to protect EDS from these attacks as it identifies and qualifies the impacts of vulnerabilities in order to develop more secure systems. Such a technique is well-suited for EDS as it evaluates the current system state, identifies system vulnerabilities, and qualifies the potential impact of threat and attack vectors, thereby providing cohesive information about a variety of risks throughout the system. As such, mitigation and prevention efforts can be better informed, ultimately resulting in more secure EDS. For example, the *Roadmap to Achieve Energy Delivery Systems Cybersecurity* published by the Energy Sector Control Systems Working Group [7], contains a dedicated section about risk analysis and assessment. It explains the need for methodologies that evaluate system state and qualifies system risk using reliable security metrics, in order to aide in decision making and mitigation processes.

However, performing risk analysis is difficult due to the innate complexity of EDS, as they are diverse systems with many heterogeneous and legacy components. As a result, there is a lot of room for missed vulnerabilities, and developing accurate risk assessment measures that can not only pick up these unknown vulnerabilities but also properly qualify them in real-time is indeed a seemingly indomitable task [11]. Despite their complexity, most EDS systems architecturally look very similar, providing a great opportunity for collaborators to work *together* to solve the aforementioned challenges. Explicitly, the following goals need to be realized:

- a way to accurately qualify the risk to an EDS based on fine-grained metrics throughout diverse aspects, levels, and components,
- an approach for identifying and mitigating security vulnerabilities,
- a live, up-to-date view of the system, its state and its current risk levels, and,
- a platform allowing collaborative decision making about securing EDS.

In order to meet these goals, we present *ExSol*, a live, real-time risk assessment ecosystem that uses collaboration, fine-grained metrics from diverse granularities of the system, and system interdependencies in order to qualify the amount of risk there is to the entire EDS. *ExSol* works by comparing the potential for *Exploitation*, i.e. threats and attack vectors, versus the implemented *Solutions*, i.e. security features and requirements, in order to understand how much risk the system may contain. Explicitly, *ExSol* uses metrics from diverse aspects of the system to provide an up-to-date view of EDS instances that allows for the understanding of system state, current threats, attack vectors and vulnerabilities, thus enabling the identification of vulnerabilities and the accurate qualification of system risk. In addition, *ExSol* leverages expert and collaborative input to determine how best to protect and secure EDS as well as to allow for the customization of the risk calculation framework to enable its utmost applicability for specific user needs.

With this in mind, this paper makes the following contributions:

(1) we introduce an approach to accurately model real-life EDS, including their interdependencies and functional relationships, based on standardized descriptions contained in a set of well-reputed regulatory documents,

(2) we provide an approach for modeling, querying, and retrieving relevant security requirements about EDS leveraging a well-defined representation in the form of ontologies,
(3) we provide a risk calculation approach at single asset and system-wide levels, which intelligently leverages both the EDS system models as well as the ontological representations mentioned above,
(4) we present an Ecosystem that facilitates collaboration and combined decision making about risk metrics and mitigation efforts, and
(5) we provide evidence of the validity, applicability, and usefulness of our approach by resorting to a series of experiments using a simulated infrastructure and case scenarios based on real attacks to EDS infrastructures.

This paper is organized as follows: we start by reviewing some important background topics and previous work in Section 2. Next, we present our approach in Section 3, provide experimental evidence of the validity of our approach in Section 4 and conclude the paper in Section 5.

2 BACKGROUND & RELATED WORK

2.1 Risk Definition and Metrics.

For the purposes of this paper, risk can be defined as the probability that a particular threat will exploit a particular vulnerability of a system [23]. Conversely, a vulnerability can be defined as the probability that a threat event will become a loss event. [6]. We expand this definition to include the impact of security requirements as well as the potential impact of security implementations. In this case, risk can be determined by comparing the assortment of attack vectors and vulnerabilities of a system with the potential security measures that may counteract such threats. Risk can be also quantified using security metrics at a variety of granularities throughout a system. For example, in the work of Lekkas and Spinellis [17], *vulnerability scorecards* that are composed of metrics that quantify good and bad aspects of a system through a goal-question-metric technique are used to understand system risk. Our approach is partly inspired by the conceptual abstraction of using metrics in order to identify "good" and "bad" scores for risk quantification, but we use different sets of metrics developed *collaboratively* to compose Exploitation (negative) and Solution (positive) subscores that are used in different ways, as explained in greater detail in Section 3. Moreover, instead of compiling metrics into a scorecard that are manually identified, we intelligently *compare* our Exploitation and Solution measures in order to quantify risk for an entire EDS system. In addition, according to Lee et al. in [16], risk quantification is based on the compilation of asset criticality, interdependence of risk factors, (dependencies between components, threats, vulnerabilities and security measures), requirements coverage, and sufficiency of conditions between risk factors. In our approach, we use a variety of metrics to understand the impact of threats, attacks, and security requirements, that are different from and beyond those used by [16]. In this case, we are relying on the concept of risk that associates threats and vulnerabilities, compared to the potential security measures that may counteract such an issue.

2.2 Risk Assessment and Analysis.

As mentioned in Section 1, risk analysis has been proposed as a valuable way to evaluate and mitigate potential attacks to EDS. In such a context, Capodieci et al. developed MICIE [1], an online alert system that evaluates the risk of EDS in real-time by detecting unexpected events and then generating risk scores based on interdependencies and functional impacts of the event. The system is monitored, and whenever an unexpected event occurs, an alert is generated and a risk score based on system component interdependence and predicted functional impact to the system is sent to operators. Additionally, Cruz et al. developed CockpitCI [3], a detection system that monitors

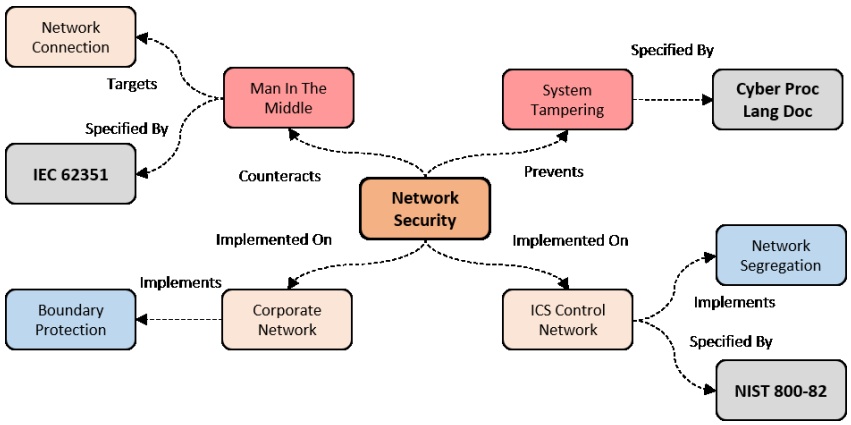

Fig. 1. An exemplary ontology for protecting EDS network infrastructures as depicted by the *OntoEDS* tool: the assets *Corporate* and *ICS Control* Networks may be targeted by the *Man In The Middle* attack and *System Tampering* threat, which are described by the IEC 62351 and Cybersecurity Procurement Language documents. Such threats may be in turn counteracted by the *Boundary Protection* and *Network Segregation* security techniques, as specified by the NIST 800-82 document.

and detects live threats within an EDS system and then uses this information to model risk using decision making processing modules. This approach uses more of an intrusion-detection strategy, in which a variety of indicators around each system component monitor the component and predict threat and propagation levels, and, along with cybersecurity information, are fed into a modeling process that estimates risk to the component. MICIE uses a system-modeling approach, and relies on the model to identify unexpected events within the system based on previous measurements, whereas CockpitCI relies on intrusion detection mechanisms to identify potential threats. As a result, these approaches may not pick up all threats within a system, if the model fails to recognize new measurements it may not have yet seen (and therefore cannot monitor for). In contrast, our approach can handle vulnerabilities that may be missed, as security metrics at a variety of levels are continuously monitored in order to quantify risk. Moreover, as it will be shown later, our approach leverages additional security metrics beyond only interdependence. Finally, our approach also takes into account metrics that quantify security countermeasures and their implementations and effectiveness, and not just the potential threats and attack vectors.

2.3 Ontology Modeling for EDS.

OntoEDS [13] is a collaborative tool that models security requirements into a comprehensive, EDS-specific ontology, allowing for stakeholders and security experts in the EDS field to model and understand security requirements, their interdependencies and their specific implementations, as well as retrieve and synthesize these requirements in an easy to use and tailored manner. *OntoEDS* also establishes a solid foundation of which additional tools can be built off of to help implement, analyze, and evaluate such security requirements. Currently, *OntoEDS* models the security requirements contained in 7 major EDS security documents from diverse organizations including the Cybersecurity Procurement Language for Energy Delivery Systems [5] developed by the Energy Sector Control Systems Working Group (ESCSWG), the National Institute of Standards and Technology (NIST) 800-82 Special Publication [21], the North American Electric Reliability Corporation Critical Infrastructure Protection (NERC CIP) standards [18], the NISTIR 7628 document [20], the Institute of Electrical and Electronic Engineers (IEEE) C37 standards [8], and the

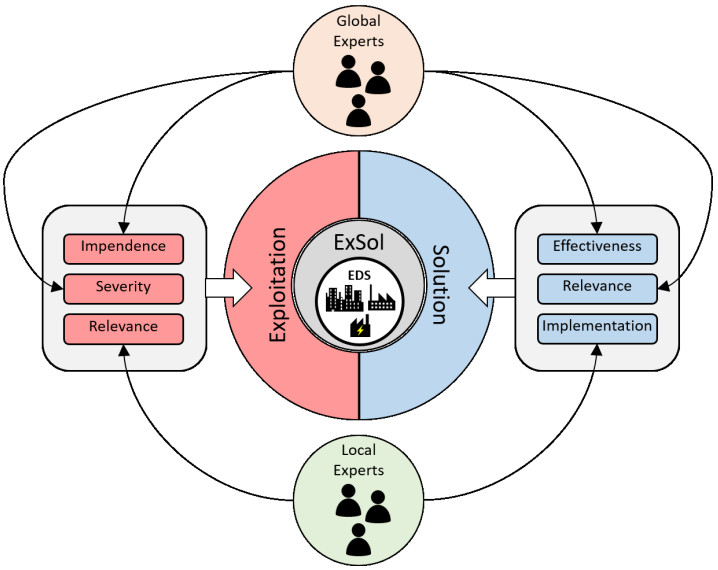

Fig. 2. A graphical depiction of *ExSol*: an overall risk score is calculated by comparing Exploitation and Solution scores. Exploitation scores are made up of *Impendence*, *Severity*, and *Relevance* metrics, and Solution scores are made up of *Effectiveness*, *Relevance*, and *Implementation* metrics. *Global* and *Local* Experts in EDS infrastructures are in charge of providing scores for each of these metrics.

International Electrotechnical Commission (IEC) 61850 and 62351 standards [9]. The ontology comprises more than 300 pages 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). An exemplary ontology depicting the protection of EDS networks is featured in Fig. 1.

3 *EXSOL*: COLLABORATIVELY ASSESSING RISKS FOR EDS INFRASTRUCTURES

As mentioned in Section 1, a way to collaboratively risk within EDS, identify and address vulnerabilities and evaluate the system state in real-time is definitively needed. To this end, we present *ExSol*, a risk assessment ecosystem that uses collaborative feedback, requirements and fine-grained metrics from diverse parts of the system to qualify the amount of risk there is to an EDS.

ExSol, graphically depicted in Fig. 2, works by comparing the potential for threats and attack vectors targeting the system (i.e., *Exploitation* scores,) and the set of security features and requirements that protect a system (i.e., *Solution* scores,) in order to understand how much risk a system or asset (specific EDS component) may contain. Exploitation and Solution scores are in turn composed of sub-metrics that elucidate specific characteristics of the security, requirements, threats or attacks for an asset, within the context of the system. These metrics include factors such as impact, relevance, effectiveness, and implementation level, and will be explained in greater detail in the following sections. In this way, by using metrics to qualify the Exploitation and Solution scores, and then comparing the two (Exploitation vs Solution), an understanding of how well protected, or, how unprotected the system may be against threats and attacks, may be attained.

Our *ExSol* methodology is generally described as follows:

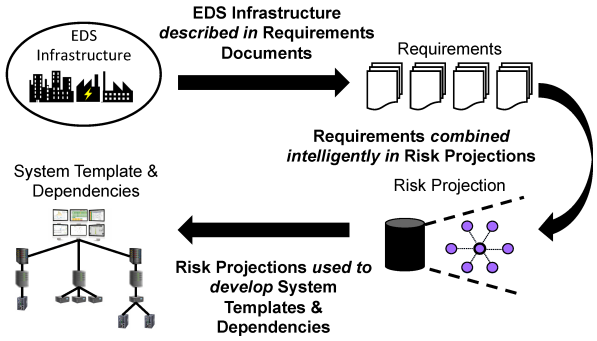

Fig. 3. System Modeling. Real systems are described in Requirement Documents, which are combined intelligently into *Risk Projections* and used to develop *System Templates* and understand *System Dependencies*.

- First, real EDS infrastructures are modeled in System Templates using characteristics of actual EDS instances and requirements from reputable organizations (such as IEC, NERC, IEEE, NIST, etc.) that are combined in intelligent and useful ways, in order to facilitate risk qualification. Additionally, within these System Templates, the *functional* and *security* dependencies are qualified, such that the impact of asset relationships with one another can be taken into consideration when calculating risk. For instance, functional relationships between Master Terminal Units (MTUs) and Programmable Logic Controllers (PLCs) such as data and command communication flows between the two are contained in the templates.
- Next, *ExSol* scores for a single *asset*, such as a MTU, can be qualified. Using the system templates and specific sub-metrics that qualify aspects of the threats, attacks, security and requirements for the asset, Exploitation and Solution scores are calculated. In must be noticed that the purpose of the scores is to rank and compare risk for an asset or within a given system, and not on the minute qualification of the *value* of risk, as might be considered under other domains, i.e., financial constraints.
- Then, in order to calculate system wide risk, or system-wide *ExSol* scores, asset *ExSol* risk scores and interdependencies between assets are multiplied with each other in order to gain an understanding of the overall *system* risk. Multiplicity of the score is useful to properly qualify the effect of exploitations and solutions. For example, as more assets have increasingly more risk, e.g., higher exploit scores, the magnitude of the final score increases (because of the multiplication), which effectively captures the more extreme (risky) state of the system. Because of such a high score, the solution countermeasures must also be equally strong, e.g., high in number.
- Finally, the specific sub-metrics and acceptable final *ExSol* system or asset scores may be collaboratively determined using our *ExSol* Ecosystem.

To explain our methodology in this section, we will first elucidate how real EDS infrastructures are modeled within our approach in order to qualify risk, next describe how our *ExSol* risk calculation occurs for a single asset and then for systems, and finally explain how the collaborative *ExSol* Ecosystem works.

3.1 System Modeling

3.1.1 Characteristics of EDS Instances. The first step in our approach included the construction of an abstract model that can accurately capture the topology and architectural design of EDS instances for risk qualification. For such a purpose, we relied on the observation of three key

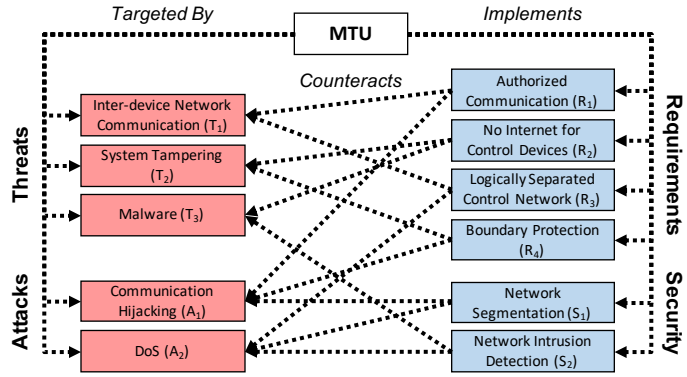

Fig. 4. Example Risk Projection for an MTU. The Threats (T_n), Attacks (A_n), Requirements (R_n), and Security (S_n) are applicable to the MTU asset, along with the pairing between threats/attacks and requirements/security. For instance, the requirement Authorized Communication (R_1) counteracts the threat Inter-device Network Communication (T_1) and the attack Communication Hijacking (A_1).

characteristics of EDS instances: First, within real-life EDS infrastructures, a system is essentially a set of assets. Second, these assets have functional relationships with each other, indicating what types of interaction specific assets may have during normal operation (such as who communicates with whom). Third, as explained in Section 1, real EDS instances in the field, although heterogeneous and component diverse, generally have similar architectures. The System Modeling steps related to these characteristics are depicted in Fig. 3 and broken down below.

3.1.2 Modeling Security Requirements. In order to tackle the security issues described in Section 1, reputable organizations within the EDS community have published a series of documents detailing architecture, security requirements, and best practices for EDS infrastructures. For example, the NIST 800-82 document [21] describes EDS instances and their functional relationships, including what assets control or send data to each other. For the purposes of our *ExSol* approach, we leveraged the *OntoEDS* tool [13], previously described in Section 2.3, which provides a well-defined ontological representation containing a variety of requirements that elucidate system configurations and security features for EDS, such as information about threats and attacks that target specific assets, and information about what security features or requirements protect those assets against what attacks and threats.

3.1.3 Risk Projections. For the purposes of risk assessment, information about security requirements and the functional relationships of EDS is crucial to properly qualify the risk to an asset based on their potential attacks, threats, and security solutions. Ontology Projections [15] allow for the quick, easy, and intelligent retrieval of sets of requirements from an ontology based on specific properties or relationships they may have. Using user-specified criteria, projections traverse the relationships between entities within an ontology in order to pull out sets of concepts and their relationships, enabling requirement understanding and analysis for the user. As an example, *OntoEDS* includes four types of projections that retrieve specific security-related goals, e.g., what security measures are needed to protect a certain asset. However, these projections were not sufficient for the needs of *ExSol*, and a *new*, more comprehensive type of projection was needed. Explicitly, we needed to take advantage of the requirements and their relationships within the ontology about how assets, threats, attacks, security features, and requirements relate to one another.

With this in mind, we leveraged the ontology projections offered by *OntoEDS* and developed our own *Risk Projections*, which use other smaller projections included in *OntoEDS* to pull out related requirements for an asset, useful for risk qualification. Risk Projections pull out functional relationships about what system components the asset is related to (for example, the relationships "communicates with," or "sends data to"). In addition, these projections find all of the threats, attacks, security features and security requirements that are related to that asset, as shown in Fig. 4. Risk Projections also pair the threats/attacks with security/requirements in order to understand what requirements may counteract what threat or attack types for that specific asset. This pairing is integral to how *ExSol* works, and is done automatically using the relationships between requirements/security and threats/attacks. For instance, as in Fig. 4, the relationship "counteracts" is used to identify the pairing of these entities. Each set of threats (T), attacks (A), requirements (R), and security (S) is stored in a 4-tuple: <T, A, R, S>. An asset may have many quad-tuples, based on the different combinations of T, A, R, and S related to an asset. These tuples are stored inside so-called *Asset Objects* in our implementation.

3.1.4 System Templates. By using the Risk Projections just described, we are able to develop abstract model representations, i.e., System Templates, for the EDS instances we wish to qualify risk for. Since EDS are defined as a set of assets, we implemented our templates as graphs that contains *Asset Objects* as nodes, and the functional relationships between them as edges (or links). An example is shown in Fig. 5. These templates contain a predetermined view of the system and their functional relationships, but can be expanded or changed by practitioners to match exactly what a real EDS infrastructure may look like.

For example, a real system modeled by further extending the System Template shown in Fig. 5 may include the assets: Programmable Logic Controllers (PLCs), Master Terminal Units (MTUs), Controllers, and a Human Machine Interface (HMI), which are modeled as nodes in the original template. Within the refined System Template, PLC_1 has a link to MTU_1 (because it communicates with the MTU), MTU_1 has a link to $Controller_1$ (since they communicate), and $Controller_1$ has a link to HMI_1 (since they communicate). However, a real EDS infrastructure may only have PLC_1, MTU_1 and HMI_1. As a result, the template can be later modified to not include $Controller_1$ anymore.

3.2 System Dependencies

3.2.1 Dependency Definition. In addition to modeling how EDS instances are architecturally set up, our proposed System Templates also contain information about the security-related dependencies between assets within a system, which are used to understand how the risk of a specific asset may be impacted by the other assets it has functional relationships to. We define a security-related dependency as the following: *if an asset* A_1 *is compromised, and another asset* A_2 *is dependent on* A_1*, then* A_2 *may be also compromised.* When a device is *compromised* this indicates that the functional operation of the device has been inhibited or changed. Moreover, such a dependency relationship indicates a *security* impact on the dependent component: i.e., if the parent asset's (A_1) security is impacted by a threat or attack, then the dependent child component (A_2) will also have a security impact, even if that component was not directly affected or targeted by the attack or threat. This relationship is represented as the child node –> parent node, such that child node "is dependent on" (–>) the parent node. There are two *types* of dependency relationships we define within *ExSol*: Command and Data Dependencies, illustrated in Fig. 6. These two types originate from the requirements summarized within our Risk Projections, as there are only two types of functional relationships between assets: either they send/receive *commands* or they send/receive *data*. Command and Data Dependencies are differentiated based on the *direction* of the child-parent relationship.

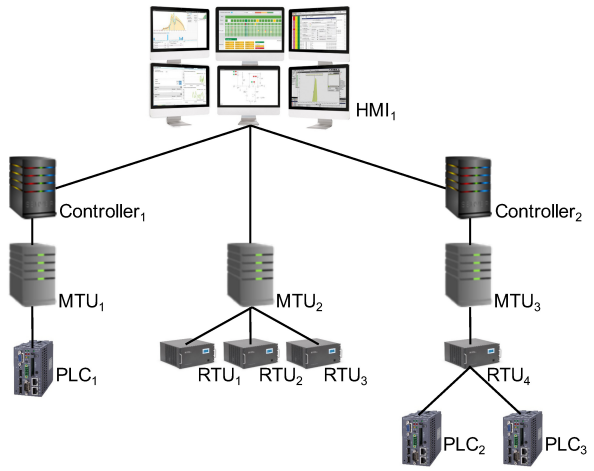

Fig. 5. A System Template depicting our running example, developed using Risk Projections that contain requirements about component relationships from the NIST 800-82 document [21], among others. Functional relationships are shown by the black solid lines connecting assets.

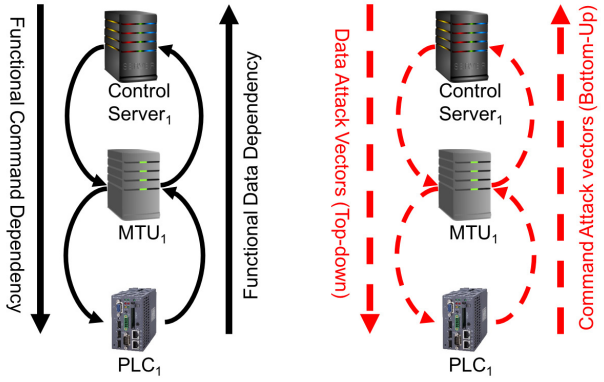

Fig. 6. A depiction of the command and data dependencies between our assets. *Functional Dependencies*, as shown in the black solid lines in the left, indicate a functional relationship between the parent and child node. *Security Dependencies* are inversely related to functional dependencies, as shown by the red dashed lines in the right. Command dependencies indicate that the child node is dependent on the parent node because the parent node sends commands to the child, and if the parent is compromised it may cause an adverse security effect to the child node. Data dependencies indicate that the child node is dependent on the parent node because the child receives data from the parent, and if the parent is compromised it may send bad data to the child, resulting in adverse effects to the child.

3.2.2 Functional Dependencies. Functional dependencies indicate *functional* interactions, e.g., sending/receiving commands/data, between two parent-child assets. In this case, the parent node has a functional relationship with the child node, and the direction of the arrow indicates which way information is being sent, *from* the parent and *to* the child, (parent−>child). These are shown in the black solid lines in Fig. 6. Functional Command Dependencies indicate that the parent asset sends *commands* to the child component, and Functional Data Dependencies indicate that the

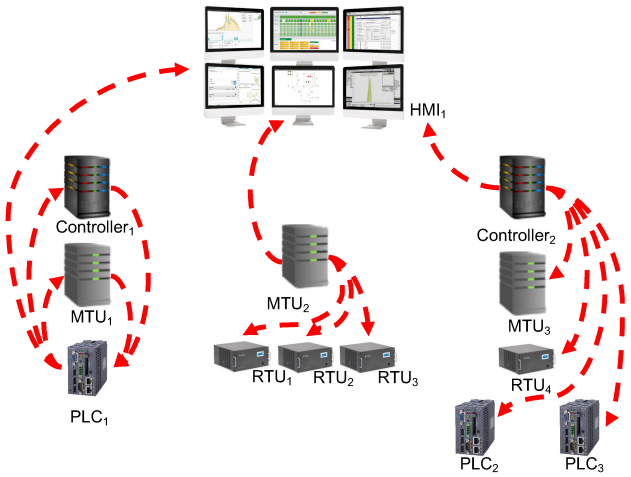

Fig. 7. Command and Data dependencies within our running example system template. For simplicity of the diagram, we show dependencies for only PLC_1, MTU_2, and $Controller_2$. Command dependency arrows point bottom-up and Data dependency arrows point top-down.

parent asset sends *data* to the child asset. For example, as in Fig. 6, the parent Control $Server_1$ issues commands to MTU_1 such as to change the MTU state, to update configurations of the MTU or to set specific thresholds. Moreover, MTU_1 is the parent of PLC_1, and it sends commands telling the PLC what to do. Going the other direction, we can see that data is sent the opposite way. Starting at the bottom, PLC_1 sends data to MTU_1 and MTU_1 sends synthesized data (including measurements from PLC and RTU devices, status and event information) to Control $Server_1$.

3.2.3 Security Dependencies. Alternatively, *security* dependency relationships are inversely related to functional relationships, with the arrow indicating that the child node is dependent on the parent node (child−>parent). These are shown in the red dashed lines in Fig. 6. For Command Dependencies, the child node is *dependent* on the parent node because it receives commands and instructions from the parent and may be adversely affected if the parent is compromised. For example, as shown in Figure 6, PLC_1 is dependent on MTU_1. If the parent node (in this case the MTU) were to be compromised, such as by a Command Injection attack, it may start sending abnormal commands to the PLC, which may result in damage to the PLC itself. For Data Dependencies, the child node is *dependent* on the parent node because it receives data from the parent node, which may cause adverse effects to the child if the data is corrupted. As shown in Fig. 6, MTU_1 is dependent on PLC_1 because the MTU receives data from the PLC. In this instance, if the PLC were to become compromised, i.e., by a "Ladder Logic" attack [24], bad data may be sent to the MTU, resulting in damage to the MTU, such as the MTU changing its alarm thresholds or other configurations. As shown in Fig. 6, Data Dependencies (from data attack vectors) are calculated in a *Top-down* method, whereas Command Dependencies (from command attack vectors) are calculated in a *Bottom-up* fashion. Some examples of command and data security dependency relationships within our Running Example System Template are shown in Fig. 7.

3.2.4 Dependency Matrix. Leveraging the definitions just presented, and in order to qualify the impact of dependencies between EDS assets, we developed a *Dependency Matrix* inspired in part by the matrix that qualifies availability developed by Chen et al. [2]. This matrix allows the security relationships between all assets within a system to be qualified by specific factors. As described

Table 1. Dependency Matrix

CN–>PN	PLC	RTU	MTU	Controller	HMI
PLC	N/A	1	5	4	5
RTU	N/A	N/A	5	4	5
MTU	3	3	N/A	5	5
Controller	2	2	3	N/A	5
HMI	N/A	N/A	N/A	N/A	N/A

later in Section 3.4, the matrix is useful in calculating system-level *ExSol* scores, where factors are applied to assets based on their dependencies before combining asset scores and calculating the final system score. This matrix is established collaboratively by the EDS community, using our Risk Projections as guidance about the dependencies between assets. Because we are qualifying security dependency relationships between assets, (which are independent of attack and threat vectors,) the matrix is constant and does not change based on attack/threat types.

An example Dependency Matrix, based on our System Template as shown in Figure 5, is shown in Table 1. For illustrative purposes, a pre-defined scale of 1 to 5 (with 1 being the weakest relationship and 5 being the strongest) is defined for the rest of this paper. N/A cells indicate there is no dependency relationship between those two assets. Child nodes are listed vertically, and parent nodes are listed horizontally within the table. For instance, when the child PLC is dependent on the parent RTU (PLC->RTU), the impact is 1 unit. Command security relationships are in the open cells, and Data security relationships are in the darker gray cells. In this case, the matrix was developed by using our Risk Projections to understand and qualify the relationships between EDS system components. From there, we developed the actual units within each cell of the table, and later refined them based on our experiments, as described in Section 4.

3.3 *ExSol* Calculation for a Single Asset

As mentioned previously, *ExSol* relies on the matching of Exploitation (attack and threat) and Solution (requirement and security) scores in order to the risk of an asset. As described in Section 2, qualifying risk relies on the use of metrics, and for *ExSol*, metrics are used to qualifying these Exploitation and Solution scores. Sub-metrics that compose Exploitation and Solution scores elucidate aspects of the security/requirements or threats/attacks of an asset within the context of the system. These metrics mathematically qualify aspects of Exploitation and Solution scores, enabling customization and allowing for the scores to be consistent amongst different EDS infrastructures. Table 2 shows the breakdown of metrics used in calculation for Exploitation and Solution scores.

3.3.1 Exploitation Scores. Exploitation scores are composed of metrics that characterize the likelihood, applicability and impact of threats (T) and attack vectors (A) on an asset, and include the sub-metrics Impendence, Severity and Relevance. As shown in Table 2, Impendence qualifies the likelihood of a threat or attack being performed on the specific asset, Severity qualifies the impact of such a threat/attack, and Relevance qualifies how applicable such a threat/attack is to the asset. These metrics all qualifies different aspects of the threat or attack, allowing for the accurate and comprehensive qualification of the Exploitation scores.

3.3.2 Solution Scores. Conversely, solution scores are composed of Effectiveness, Relevance and Implementation sub-metrics that qualifies the effectiveness, applicability and level of implementation of Requirements (R) and Security (S) for an asset. As shown in Table 2, Effectiveness the perception on the ability of the requirement to deter an attack or threat, Relevance the applicability

Table 2. *ExSol* Sub-metric explanations.

Score	Metric	Definition	Defined By
Exploitation	Impendence (T_i or A_i)	Likelihood/Frequency of threat being exploited or attack being performed.	Global Expert
Exploitation	Severity (T_s or A_s)	Impact and damage of threat/attack on the asset.	Global Expert
Exploitation	Relevance (T_r or A_r)	How applicable or targeted to the asset the threat/attack is.	Local Expert
Solution	Effectiveness (R_e or S_e)	Perception on the ability of the requirement to deter/counteract an attack or threat.	Global Expert
Solution	Relevance (R_r or S_r)	Applicability of a requirement to the asset being analyzed.	Global Expert
Solution	Implementation (R_i or S_i)	Perception on the effectiveness of the implementation of a given requirement in the system.	Local Expert

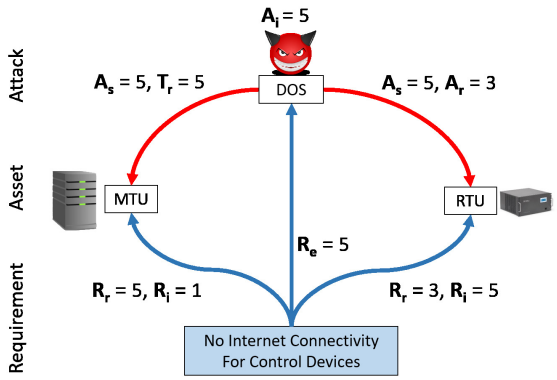

Fig. 8. An example of Exploitation and Solution Sub-metrics for the assets MTU and RTU, the attack vector DoS and the requirement No Internet Connectivity for Control Devices, using our example scale from 1 (low) to 5 (high).

of the requirement to the asset, and Implementation the perception on the effectiveness of the implementation of a specific requirement for an asset. These metrics different aspects of requirements and security, thereby allowing for the accurate and comprehensive qualification of the Solution score. Example metrics for an MTU are shown in Tables 3 and 4.

For the rest of this paper, each metric's score is calculated on a scale from 1 (least) to 5 (greatest), following the example depicted in Table 1. However, implementations of our *ExSol* approach can define their own scales based on their own customization needs.

3.3.3 Addressing our Running Example. An example of how these metrics are quantified is shown in Figure 8, using our example scale (from 1 to 5). In this example, we have two assets, an MTU and an RTU, the attack DoS, and the requirement "No Internet Connectivity for Control Devices".

Table 3. Example ExSol Asset Calculation - T/A Metrics for an MTU

	Inter-device Net. Comm. (T_1)	System Tampering (T_2)	Malware (T_3)	Comm. Hijacking (A_1)	DoS (A_2)
Impendence	4	3	4	4	3
Severity	4	2	4	5	5
Relevance	5	2	3	5	5
Sub-score	**80**	**12**	**48**	**100**	**75**

Table 4. Example ExSol Asset Calculation - R/S Metrics for an MTU

	Authorized Comm. (R_1)	No Internet for Control Devices (R_2)	Logically-Separated Control Network (R_3)	Boundary Protection (R_4)	Network Segmentation (S_1)	Network Intrusion Detection (S_2)
Effectiveness	5	4	4	3	4	4
Relevance	5	5	4	3	4	3
Implementation	4	5	4	5	5	4
Sub-score	**100**	**100**	**64**	**45**	**80**	**48**

Starting at the DoS attack, we see that its Impendence score (A_i) is high at a 5, as this attack is very likely to be performed on the two assets. The DoS vector has a level 5 severity (A_s) for both the MTU and the RTU, as the damage it performs upon successful attack is very high. However, the attack's relevance (A_r) is a 5 for the MTU, but is only a 3 for the RTU because the attack is less targeted towards an RTU but highly targeted for an MTU. On the bottom, the requirement has a level 5 effectiveness (R_e) against the attack vector, meaning it is effective at preventing DoS attacks. The requirement has a 5 level relevance (R_r) for the MTU as it is very applicable to the asset, but has only a 3 relevance for the RTU, as it is less applicable. Finally, the implementation score (R_i) of the requirement is a 1 for the MTU, perhaps because the MTU still connects to the internet to communicate with the local control server, whereas the implementation score is a 5 for the RTU because it has correctly implemented the requirement, and does not connect to the internet whatsoever.

3.3.4 Assigning Scores to Sub-metrics. As sub-metrics are an integral part of the qualification for Exploitation and Solution scores, how these metrics are evaluated and determined is an important consideration. As such, in our approach, these sub-metrics are developed in a collaborative fashion by types of experts in EDS infrastructures, denoted as *Global* and *Local* Experts, as shown in Fig. 2. Global Experts *collaboratively* define the metrics that are less dependent on actual implementations within the system, and instead are more based on information about the attack, threat, security or requirement itself. Specifically, these include Impendence and Severity for the Exploitation score and Effectiveness and Relevance for the Solution score. These metrics can be adapted by the community based on changes in the threatscape, or the attainment of new knowledge (such as about security/requirement criticality or importance). This process is facilitated by our *ExSol Ecosystem*,

explained in greater detail below in Section 3.5. The potential arbitrariness or subjectivity of these Global scores is mitigated by the fact that these scores are determined collaboratively and agreed upon universally, such that they are not changing amongst different people or systems: these metrics form a standardized set of measures applicable across diverse EDS infrastructures. It is also important to note that the focus of *ExSol* is on an *asset*, which facilitates its compatibility and consistency amongst multiple systems. Alternately, Local Experts *individually* determine the local scores that aspects of their particular system. In this case, these measures characterize implementation-level information for the asset. These metrics include Relevance for the Exploitation score, and Implementation for the Solution score.

3.3.5 ExSol Asset Calculation Algorithm. With all this in mind, we now present how to calculate the actual *ExSol* scores for individual assets. As mentioned previously, to risk for an asset Exploitation scores and Solution scores are matched up against one another to form the final *ExSol* asset score. To explain this calculation algorithm, we present an example to calculate the asset ExSol score for an MTU using the Tables 3 and 4, developed as examples.

(1) First, our Risk Projection is used to create *Asset Objects* that contain all Threats (*T*), Attacks (*A*), Requirements (*R*), and Security (*S*) related to a given asset, and the relationships between each of these entities. For example, as shown in Table 3, 4, and Fig. 4, we identify the following set of entities for an MTU: *Threats:* Inter-device Network Communication (T_1), System Tampering (T_2), Malware (T_3); *Attacks:* Communication Hijacking (A_1), DoS (A_2); *Requirements:* Authorized Communication Between Control Devices (R_1), No Internet For Control Devices (R_2), Logically Separated Control Network (R_3), Boundary Protection (R_4); *Security:* Network Segmentation (S_1), Network Intrusion Detection (S_2).
(2) The Risk Projection next creates tuples in the form <T, A, R, S>, pairing related entities based on their relationships for the asset. For the MTU in our running example, 10 tuples are developed, such as the tuple <T_1, A_1, R_1, S_1>. This tuple indicates that the *Threat* Inter-device Network Communication (T_1) may be realized as the *Attack* Communication Hijacking (A_1) and may be counteracted by the *Requirement* for Authorized Communication Only Between Control Devices (R_1) which may be implemented through the *Security* feature Network Segmentation (S_1).
(3) Sub-metric scores for each of the identified (T, A, R and S) entities are then determined by Global and Local Experts, as mentioned before. Global sub-metric scores may be previously identified through collaboration in the Ecosystem, as it will be discussed in Section 3.5, and local scores may be determined by Local Experts within their own implementations. In our running example, the sub-metric scores for each of the entities (T, A, R and S) surrounding the MTU are shown in Tables 3 and 4.
(4) Next, for each tuple, the Exploitation sub-scores for each T and A, and the Solution sub-scores for each R and S are calculated. Equation 3.1 (1) and (2) are used for Threat and Attack metrics respectively and Equation 3.2 (3) and (4) are used for Requirement and Security metrics respectively.

Definition 3.1. Exploitation Sub-score Calculations:

$$(T) = T_i * T_s * T_r \tag{1}$$

$$(A) = A_i * A_s * A_r \tag{2}$$

where T_i, T_s, T_r, A_i, A_s, and A_r stand as defined in Table 2.

Definition 3.2. Solution Sub-score Calculations:

$$(R) = R_e * R_r * R_i \tag{3}$$

$$(S) = S_e * S_r * S_i \tag{4}$$

where R_e, R_r, R_i, S_e, S_r, and S_i stand as defined in Table 2.

Essentially, the sub-metrics for the T, A, R, and S entities are multiplied together. This is done in order to the overall effect of each Threat, Attack, Requirements and Security measure. Once aspects (in this case, specific metrics), are used to the entity, these metrics have to be combined to gain an overall understanding of the strength of the entity. For example, the T_1 subscore is calculated as: $T_1 = T_i * T_s * T_r = 4 * 4 * 5 = 80$. In another instance, the R_1 subscore is calculated as: $R_1 = R_e * R_r * R_i = 5 * 5 * 4 = 100$.

(5) After the Exploitation and Solution sub-scores are calculated, the final asset *ExSol* score is calculated using Equation 3.3 (5) for each tuple.

Definition 3.3. *ExSol* Score Calculation:

$$ExSol = (R * S) - (T * A) \tag{5}$$

In this equation, the Solution sub-score (Requirement and Security scores) are combined and subtracted against the Exploitation sub-score (combined Threat and Attack scores). This is done in order to determine the amount of risk the asset has. By comparing the strength of the security and requirements *protecting* the asset against the strength of the attack and threat vectors *targeting* the asset, we can understand the amount of risk that asset has. Ideally, the strength of the Exploitation score (threats and attack vectors) should not be more than that of the Solution score (security and requirements,) as this indicates the asset's security requirements are not equipped to handle the potential attacks and threats targeting that asset, and thereby that the asset is in a risky state. For our running example, the ExSol asset calculation for the first tuple is: *ExSol* = $(R_1 * S_1) - (T_1 * A_1) = (100 * 80) - (80 * 100) = 0$.

3.3.6 Understanding ExSol *Scores.* Finally, risk of the asset may be evaluated based on the set of *ExSol* asset scores developed from all tuples. In this case, we can leverage either the tuple with the most negative asset *ExSol* score (indicating that attack/threat is not well counteracted by the implemented requirement and security for the asset), or the average of all of the tuple *ExSol* scores. For instance, in our MTU calculation example shown in Fig. 9 (2), we can determine its riskiest score is -4400, and that it has an average *ExSol* score of about -613. Overall, the final *ExSol* asset scores can be understood as follows: leveraging the sample metric scale depicted in our running example, an *ExSol* score greater than 0 may indicate that the asset under analysis has good security mechanisms, and ideally, the greater the *ExSol* score, the more secure the asset may be. Conversely, if the *ExSol* score is at 0, e.g., the Solution and Exploitation scores are matched, the asset may be assessed as having the minimal amount of security necessary to defend against attacks. Finally, an *ExSol* score below 0 may indicate the asset does not have enough security mechanisms (or they may not be implemented correctly) to defend against attacks. In general, the further the score is below 0, the riskier the asset is. When using different customized scales, the numerical threshold values may change, but the reasoning just described may stay the same.

3.4 *ExSol* System-Wide Approach

Now that we have a way to calculate individual asset scores, we can use these scores and the dependencies between assets to calculate system *ExSol* scores. To illustrate the *ExSol* system risk calculation, we present an example using our System Template in Fig. 5. The *ExSol* system process, displayed in Fig. 9, goes as follows:

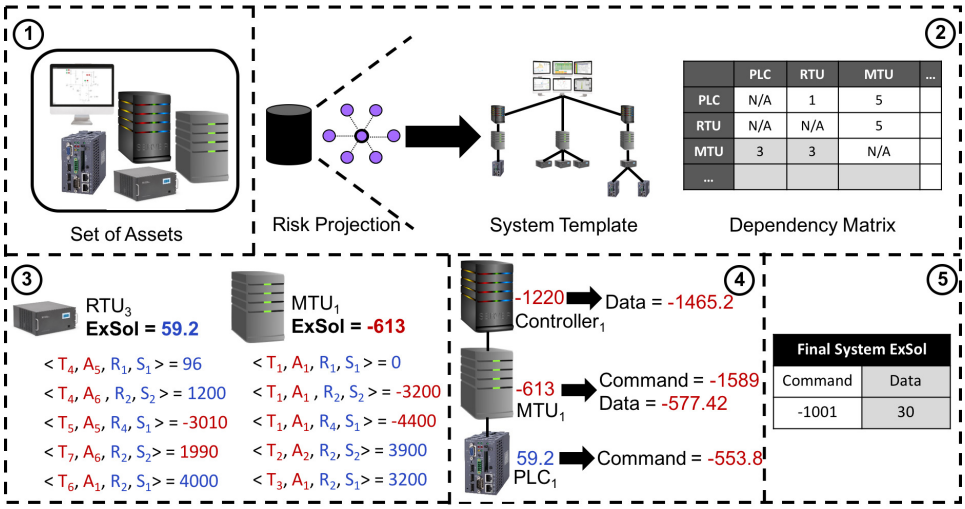

Fig. 9. *ExSol* System Calculation Steps: (1) The set of assets are gathered for the system. (2) System Templates and a Dependency Matrix are developed using Risk Projections. (3) Individual Asset *ExSol* scores are calculated. (4) Asset scores are adapted based on system dependencies using the Dependency Matrix. (5) The final *ExSol* system score is calculated by combining *ExSol* asset scores.

(1) First, we obtain the set of assets included in the actual EDS infrastructure being evaluated. In our running example, such assets include PLCs, RTUs, MTUs, Controllers, and a HMI.
(2) Next, Risk Projections for each asset are used to identify dependencies within the system and develop a System Template and Dependency Matrix. In our running example, the System Template in Figure 5 and the Dependency Matrix in Table 1 are developed. Abbreviated depictions are shown in Fig. 9 (1). For example, in terms of command dependencies, we identify that the PLC receives commands from the MTU (dependency weight of 5) and the MTU receives commands from the controller (dependency weight of 4). In terms of data dependencies, we see that the MTU receives data from the PLC (dependency weight of 3), and the Controller receives data from the MTU (dependency weight of 2). A System Template and Dependency Matrix are developed based on these relationships. More information about dependencies is explained in detail in Section 3.2.
(3) Then, *ExSol* scores are calculated individually for each asset in the system, following the steps explained previously in Section 3.3. In order to get the final *asset ExSol* score, we average the individual sets of *ExSol* scores from all of the asset's tuples. For example, within our running example in Fig. 9 (2), we calculate MTU_1's average *ExSol* as -613.
(4) Later, using the System Template and the Dependency Matrix, individual asset scores are adapted based on their security dependencies. This is done for the two types of dependencies we have within our approach (Command and Data dependencies), and correlates to the way in which the System Template is traversed, and the order in which individual asset scores are adapted. Command dependencies are traversed in a *Bottom-up* approach, in which the child node's score at the bottom is adapted based on its relationship with its parent before moving up to the next node. Oppositely, Data dependencies are traversed in a *Top-down* approach, in which the child node at the top is first adapted based on its parents (and its dependencies), before moving down to the next node. Based on the qualification of the security dependencies between assets as determined using the Dependency Matrix, a percentage of the Parent Node's (PN) score is added

to the Child Node's (CN). In this case, using our example matrix, if the dependency value is 5, 100% of the PN's score is added, if the weight is 4, 80% of the PN's score is added, and so on. Generally, this *percentage* dependency weight, denoted at %DW, can be calculated as:

Definition 3.4. Percentage Dependency Weight.

$$\%DW_i = (DW_i * 100)/MAX - SCALE \tag{6}$$

where DW is the Dependency Weight for an asset i obtained from the Dependency Matrix, and MAX-SCALE is the maximum value defined for the score scale being used.

By using percentages to qualify the Dependency Matrix values, the dependency relationships between assets are properly qualified within the system *ExSol* score.

Definition 3.5. Security Dependency Adaptions.

$$Adapted\_ExSol_i = CN_i + (\%DW_i * PN_i) \tag{7}$$

where CN is the original child node ExSol *score, DW is the Percentage Dependency Weight and PN is the original parent node* ExSol *score for asset i.*

As shown in Fig. 9 (3), for the left branch of our system including PLC_1, MTU_1, and $Controller_1$, Command Dependencies of PLC->MTU and MTU->Controller would adapt the asset scores as follows: *Adapted_ExSol* (PLC_1) = 59.2 + (100% * -613) = -553.8. *Adapted_ExSol* (MTU_1) = -613 + (80% * -1220) = -1589. For Data Dependencies of Controller->MTU and MTU->PLC, the asset scores would be: *Adapted_ExSol* ($Controller_1$) = -1220 + (40% * -613) = -1465.2, *Adapted_ExSol* (MTU_1) = -613 + (60% * 59.2) = -577.48.

(5) Finally, the system-wide *ExSol* score is calculated by taking a *conservative* approach and choosing the lowest *Adapted_ExSol* score from all the assets within the EDS instance under study. Such an approach is based on the fact that the previous calculations for *Adapted_ExSol* scores are intended to determine the impact each asset has within a given EDS instance. This way, an EDS instance can be as secure as its riskiest component. Moreover, as a result of the two types of dependency traversals described before, two final System *ExSol* scores are developed (one for Command Dependencies and the other for Data Dependencies). The *ExSol* score for Command Dependencies best determines the risk of a system related to command-based attacks that use force to take over parts of an EDS, such as a Command Injection Attack, whereas *ExSol* scores for Data Dependencies best fy the risk for data-focused attacks, such as a Ladder Logic attack. In our example in Fig. 9 (4), we come up with the following scores: Command System *ExSol* score = -1001 and Data System *ExSol* score = 30. The final system *ExSol* scores are understood in the same way as asset *ExSol* scores, as explained in Section 3.3.

3.5 *ExSol* Ecosystem and Collaboration

As mentioned in Section 1, we aim to introduce a platform for a collaborative ecosystem that supports group decision making, and assists the EDS community in understanding the state of their systems, qualifying risk using *ExSol* and mitigating any found vulnerabilities. Incentives for participation in such an ecosystem would include: improved security outcomes, better risk management, reduction of cognitive burdens and burnout for EDS practitioners.

To this end, the *ExSol* Ecosystem provides an interconnected, live, community-based score tabulation scheme that holds the latest community-established *ExSol* metrics for an asset or a set of assets (i.e., the T, A, R and S Global metrics), as it was hinted in Table 2, along with acceptable ranges that an asset's or system's *ExSol* score should be within in order to be at an acceptable state

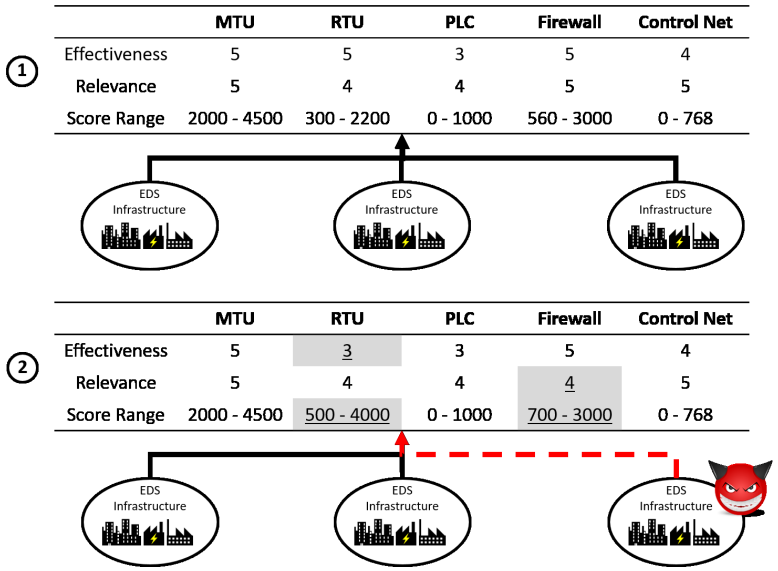

Fig. 10. A depiction of the *ExSol* Ecosystem. A set of Global metrics are determined by the EDS community (1) and can be updated based on changes in the threatscape such as a new attack type that alters the metric scores and acceptable threshold levels as in (2).

of security. Such an approach allows for the EDS Community to collaboratively decide on these scores and ranges, based on everyone's various infrastructures. As a result, common *calibrated* scores will be developed that are similar across organizations, thereby allowing accurate *ExSol* risk score calculations. Experts in the EDS community, referred as *Global Experts* in Fig. 2 and Section 3.3, may work together to update these scores based on changes in the threatscape and their expertise, in order to continue to keep them current.

3.5.1 Example Use Case. The *ExSol* Ecosystem can help collaborators stay on top of the state of security their EDS systems have, as well as respond to vulnerabilities and attacks. For instance, Figure 10 shows what a sample Ecosystem might look like. In part (1), a variety of assets are listed, along with some of their current sub-metric scores (in this case their Solution Relevance and Effectiveness metrics). In addition, there are suggested score ranges for each asset, that indicate what their acceptable *ExSol* asset scores are. For example, the MTU is considered in an acceptable state of security if its *ExSol* score is within the range of 2000 to 4500. However, in part (2), an attacker has discovered a vulnerability in the 3rd system. As a response, collaborators come together, and, using *ExSol*, determine that the RTU and Firewall security configurations have been taken advantage of. Consequently, their current sub-metrics decrease (as they are regarded as less effective at preventing attacks), and the overall score ranges for these assets increase, as they need additional security measures (and therefore higher *ExSol* scores) in order to be considered secure. This way, *ExSol* may help collaborators to respond to vulnerabilities and quickly determine what measures may be implemented to appropriately mitigate risk. Finally, it is important to note that although *ExSol* is intended to aide collaboration for the EDS community, individual stakeholders may keep their independence in regards to how they may react to the information they learn from the Ecosystem and the actual *ExSol* risk scores. Ultimately, *ExSol* allows stakeholders to *defend* EDS as a community, but *protect* systems individually.

4 EXPERIMENTAL EVALUATION

Having presented our approach in Section 3, we now aim to provide experimental evidence of its suitability to effectively assess security risks for EDS infrastructures. To this end, we developed a series of experiments within both simulated and real EDS infrastructures, which show how changes in asset metric scores are identified and result in valid changes to the overall *ExSol* score, allowing for further interpretations for the purposes of risk assessment as discussed previously in Section 3.3.

4.1 Simulation Infrastructure.

For our experiments, we developed an EDS simulation infrastructure based on the System Template as shown in Figure 5, that sends and receives commands and data, and monitors the state of each asset. The simulation uses a 48-core server running a combination of Virtualbox, OpenStack and Mininet. Multiple virtual machines (VMs) were developed in order to simulate the HMI and the Supervisory Control and Data Acquisition (SCADA) infrastructure. One VM was used as the main controller representative of a HMI, and the other VMs emulated the rest of the smart grid, including PLCs, MTUs, RTUs, and Controllers. Moreover, functional relationships were emulated between components based on the data and commands that were transmitted between them, allowing for assets within the infrastructure to send/receive commands and data and report their state to the HMI VM. This way, we were able to simulate attacks on different assets and monitor the impact of such attacks on each asset in order to inform our calculations. We also implemented our *ExSol* approach in Java, which, besides leveraging the *OntoEDS* tool for the Risk Projections discussed in Section 3.1, also receives asset state updates related to the calculation of security metrics from our simulation infrastructure, and automatically calculates *ExSol* risk scores at both asset and system levels.

4.2 Experiment 1: Changes in Infrastructure Impact

As described before in this paper, *ExSol* is intended to detect fine-grained changes in infrastructure and assets, so it can identify vulnerabilities, new threats, and subtle shifts in assets (for example, in their state, functioning or operation.) These changes are specifically at a *local level*; as a result, changes at the local implementation level for assets need to result in logical changes to the final *ExSol* score, even if global metrics stay the same. As such, our first set of experiments was intended to test the relationship between changes in our infrastructure and changes in the system, i.e. validate that changes in local infrastructure (asset state) result in changes to the overall *ExSol* score. To perform this experiment, we assumed that global scores stayed at an intermediate level (all sub-scores had a value of 3), and changed local scores only using our implemented engine. Our intuition was that local metrics resulting in low Exploitation scores and high Solution scores would result in very high (positive) *ExSol* scores, local metrics resulting in high Exploitation scores and low Solution scores would result in very low (negative) *ExSol* scores, and local metrics resulting in matched Exploitation or Solution scores (such as both being high or low) would result in a matched *ExSol* score near 0. We tested this hypothesis by changing the local metric values for Relevance and Implementation, and having our engine calculate the Exploitation and Solution sub-scores and the final *ExSol* scores. Explicitly, we compared low (value of 1), medium (value of 3) and high (value of 5) Relevance metrics (which are fed into our Exploitation scores) with low (1), medium (3) and high (5) Implementation metrics (which are fed into our Solution scores), following the scale used in our running example. Fig. 11 shows our resulting Exploitation and Solution sub-scores compared to the final *ExSol* scores, ultimately confirming our hypothesis: high Exploitation scores and low Solution scores result in low (very negative) scores, shown on the right of the figure, and, on the left, that high Solution scores and low Exploit scores result in high (very positive) scores.

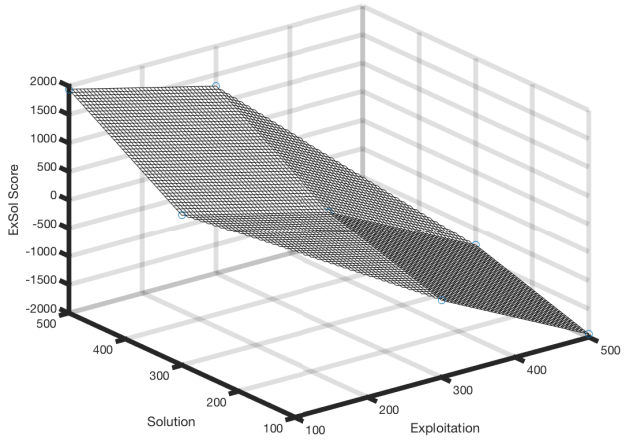

Fig. 11. Experimental results showing the distribution of *ExSol* Scores. High Solution and Low Exploitation scores result in High *ExSol* scores and Low Solution and High Exploitation scores result in Low *ExSol* scores.

4.3 Experiment 2: Attack Case Scenarios

Our next phase of experimentation was intended to test how *ExSol* reacted in real-life scenarios where both global and local metrics change. We developed a variety of real-world attack scenarios based on common vulnerabilities detailed by the U.S. Department of Energy [12] that we simulated on our infrastructure to see 1) how subtle changes in the infrastructure resulted in changes in our scores, and 2) how dependencies within the context of these attacks affect components' *ExSol* scores, i.e. how parent scores of affected components from the attack may propagate down to child components. For illustrative purposes, two of such scenarios, involving a Command Injection and Data Spoofing attack, and a Buffer Overflow attack, are elucidated next.

4.3.1 Command Injection and Data Spoofing. Assuming the system shown in Fig. 5, an attacker compromises RTU_2 and PLC_1, and begins to execute a Command Injection attack on the RTU, and a Data Spoofing attack on the PLC. As a result, the Exploitation submetrics (Impendence, Severity and Relevance) are increased for the Command Injection Attack targeting the RTU, and for the System Tampering Threat and Data Spoofing Attack for the PLC. Assuming an initial starting score for all components of 0, we can see in Table 5 (Part 1) highlighted in gray how the *ExSol* scores have changed for both the RTU and PLC. In addition, when we look at the resulting Data Dependency scores, we can see that the *ExSol* scores of MTU_1 and MTU_2 have changed, as well as $Controller_1$. This is to be expected, as these data attacks should have adverse effects on the components that are receiving data from the compromised entities (i.e. MTU_1 and $Controller_1$ who are receiving compromised data from PLC_1, and MTU_2, who is receiving compromised data from RTU_2). Next, as MTU_1 and MTU_2 are receiving bad data from the PLC and RTU, some of their configuration scales are changed. Detecting this, Exploitation scores for the MTUs are updated and reflected in their *ExSol* scores as shown in the highlighted cells in Table 5 (Part 2). Finally, EDS operators may update the security measures for the components related to Integrity Validation, Authorized Communication and Data Encryption of sent messages. As a result, the sub-metrics for these security requirements are updated accordingly, and the resulting *ExSol* scores shown in Table 5 (Part 3) are given.

Table 5. Command Injection & Data Spoofing.

	Component	Original	Updated by %DW
Part 1	**Controller$_1$**	0	-554
	MTU$_1$	0	-831
	MTU$_2$	0	-317.4
	RTU$_2$	-529	-529
	PLC$_1$	-1385	-1385
Part 2	**Controller$_1$**	0	-554
	MTU$_1$	-140	-971
	MTU$_2$	-534	-851.4
	RTU$_2$	-529	-529
	PLC$_1$	-1385	-1385
Part 3	**Controller$_1$**	0	128.4
	MTU$_1$	658	807.6
	MTU$_2$	221	221
	RTU$_2$	837	837
	PLC$_1$	321	321

4.3.2 Buffer Overflow. In this scenario, shown in Table 6, Controller$_2$ is compromised by a Buffer Overflow attack. In this case, the attacker uses vulnerabilities in Network Protocols to successfully compromise the Controller. As a result, first, as in Table 6 (Part 1), the Controller's submetrics for the Threat, Vulnerabilities in Network Protocols, are increased to represent this increased threat. As a result, the component's score greatly decreases, as do the other interdependent components of the controller, as shown in the propogation of scores in the table. Next, in Table 6 (Part 2), the Buffer Overflow attack begins to occur, and the submetrics relating to such an attack are updated, resulting in even more negative scores. In Table 6 (Parts 3 and 4), the attacker, using Controller$_2$, begins to cause DoS attacks to MTU$_3$ and RTU$_4$. The attack vector of DoS's Exploit submetrics are updated to reflect the occurrence of these attacks, and the resulting scores change as shown in the tables. Finally, in Table 6 (Part 5), an EDS operator implements some additional security measures on these components in order to protect them from such an attack. These include updating the control server's library to protect it from SQL injections, updating patches throughout the components, and ensuring securing coding practices are used as well as standard protocols. As a result, the scores increase as shown in Table 6 (Part 5). These two scenarios illustrate how minute changes within metrics as a result of attack vectors are detected and reflected within *ExSol*, validating the suitability of our approach to be deployed within real environments.

4.4 Experiment 3: Real World *ExSol* Calculation.

Our final phase of experimentation used real world data received from a partner utility grid, deployed at the Lawrence Berkeley National Laboratory in California, that controls and provides electricity for the University of California, Berkeley, and other sites. More information about the grid can be found in the work by Peisert et al. [19]. This data was used to calculate *ExSol* asset scores for Phasor Measurement Units (PMUs), specifically focused on the security requirement Intrusion Detection and anomalous PMU behavior as a result of various attack vectors (such as DoS). We received magnitude and angle measurements for voltage and current from PMUs deployed within the Berkeley grid. A sample depiction of the data is shown in Fig. 12. Similar to the first

Table 6. Buffer Overflow Attack Scenario.

	Component	Original	Updated by %DW
Part 1	**$Controller_2$**	-438	-438
	MTU_3	0	-438
	RTU_4	0	0
	PLC_2	0	-350.4
	PLC_3	0	-350.4
Part 2	**$Controller_2$**	-915	-915
	MTU_3	0	-4575
	RTU_4	0	0
	PLC_2	0	-732
	PLC_3	0	-732
Part 3	**$Controller_2$**	-915	-915
	MTU_3	-985	-1900
	RTU_4	0	-985
	PLC_2	0	-985
	PLC_3	0	-985
Part 4	**$Controller_2$**	-915	-915
	MTU_3	-985	-1900
	RTU_4	-453	-1438
	PLC_2	0	-985
	PLC_3	0	-985
Part 5	**$Controller_2$**	1337	1337
	MTU_3	192	192
	RTU_4	276	468
	PLC_2	0	55.2
	PLC_3	0	55.2

experiment, we used static global measurements and derived the local scores from the voltage and angle magnitude measurements. Following the work by Jamei et al. [10], we were able to determine the state of the PMU and its Local metrics for security/requirement Implementation and attack/threat Relevance by using voltage and current magnitude levels over time to determine if the PMU was at an anomalous state or not. We assumed all global metrics had a score of 3, and the tuple we were calculating PMU asset ExSol scores for was the Threat (T) of Anomalous Behavior, the Attack (A) of DoS, the Requirement (R) of System Monitoring, and the Security (S) of Intrusion Detection. Our results are shown in Table 7, in which the local Implementation metrics for the Requirement and Security, the local Relevance metrics for the Attack and Threat, the Solution and Exploitation sub-scores, and the final *ExSol* scores for the PMU are displayed at each time point. We only received positive measurements from the grid, with no anomalous behavior being detected, and, as expected, we calculated all positive *ExSol* scores for the PMU.

5 CONCLUSION AND FUTURE WORK

In this paper we have presented *ExSol*, a collaborative risk assessment ecosystem that uses fine-grained metrics and system interdependencies to qualify the amount of risk there is to an EDS,

```
TIMESTAMP, VOLT_MAG, CURR_MAG, VOLT_ANGL,CURR_ANGL
1483576014,120.24095,18.172830,193.29812,204.46424
1483576014,120.23758,18.197536,193.23658,204.43585
1483576014,120.23809,18.221977,193.17518,204.41525
1483576015,120.24928,18.182682,189.64651,200.79434
1483576015,120.24933,18.187477,189.59088,200.78160
1483576015,120.24862,18.178153,189.53096,200.76989
1483576016,120.25269,18.141019,186.97297,198.26708
1483576016,120.25315,18.156663,186.91018,198.16394
1483576016,120.25547,18.163524,186.85005,198.06536
```

Fig. 12. An excerpt of sample PMU data from a utility grid.

Table 7. PMU Asset ExSol Score Calculations

Time	T	A	R	S	Expl	Soln	ExSol
1	1	2	5	5	162	2025	1863
2	1	1	5	5	81	2025	1944
3	2	1	5	4	162	1620	1458
4	2	2	4	4	324	1296	972
5	1	1	5	5	81	2025	1944

allowing for collaborators to work together and make group decisions about protection and mitigation actions for their infrastructures.

ExSol is meant to be an extensible framework in which its ecosystem can be continually added to, in order to better expand and represent EDS infrastructures, as well as prove even more useful for EDS stakeholders. As a result, there are a variety of places throughout the framework which could be adapted for future goals, including individual component *ExSol* calculation, system *ExSol* calculation, and the overall *ExSol* framework. In terms of individual component *ExSol* calculation, we aim to develop an optimization process that uses *OntoEDS* and finds the best tuple pairs, in order to provide a succinct summary set of tuples that may best represent the best set of security measures for that component, as an additional tool that *ExSol* provides and that Stakeholders could use to understand and validate security implementations within their system. Moreover, the system *ExSol* approach could be expanded in a variety of ways, such as how the final score may be calculated. Future work may also investigate alternative approaches to combining the final score, as well as validating which of the methods is the best suited to various types of EDS infrastructures. Finally, even when *ExSol* does not go into low level details to examine specific flaws in the devices, future work may focus on updating *ExSol* to serve as a *zero-day* notification system, allowing for metric scores to detect adverse changes to devices, which will ultimately result in a decrease in the overall *ExSol* score, thus indicating to EDS operators that there is a potential vulnerability that needs to be addressed.

REFERENCES

[1] P. Capodieci, S. Diblasi, E. Ciancamerla, M. Minichino, C. Foglietta, D. Lefevre, G. Oliva, et al. 2010. Improving resilience of interdependent critical infrastructures via an on-line alerting system. In *Complexity in Engineering, 2010.* IEEE, 88–90.

[2] Qian Chen, Robert K Abercrombie, and Frederick T Sheldon. 2015. Risk assessment for industrial control systems quantifying availability using mean failure cost (MFC). *Journal of Artificial Intelligence and Soft Computing Research* 5, 3 (2015), 205–220.
[3] T. Cruz, J. Proença, P. Simões, M. Aubigny, M. Ouedraogo, A. Graziano, and L. Yasakhetu. 2014. Improving cybersecurity awareness on industrial control systems: The CockpitCI approach. In *13th Euro Conf. on Cyber Warfare and Sec.* 59.
[4] Dragos Inc. 2017. *CRASHOVERRIDE Analysis of the Threat to Electric Grid Operations.* https://dragos.com/blog/crashoverride/CrashOverride-01.pdf
[5] Energy Sector Control Systems Working Group (ESCSWG). 2014. Cybersecurity Procurement Language for Energy Delivery Systems.
[6] Jack Freund and Jack Jones. 2015. Measuring and Managing Information Risk. Butterworth-Heinemann, Boston, 13 – 23.
[7] Energy Sector Control Systems Working Group et al. 2011. Roadmap to achieve energy delivery systems cybersecurity. *Energetics, Inc, URL https://www. controlsystemsroadmap. net/ieRoadmap\% 20Documents/roadmap. pdf* (2011).
[8] IEEE. 2017. C37.118.1-2011 - IEEE Standard for Synchrophasor Measurements for Power Systems. standards.ieee.org/findstds/standard/C37.118.1a-2014.html
[9] International Electrotechnical Comission (IEC). 2017. Core IEC Standards. http://www.iec.ch/
[10] M. Jamei, A. Scaglione, C.n Roberts, E. Stewart, S. Peisert, C. McParland, and A. McEachern. 2016. Automated Anomaly Detection in Distribution Grids Using muPMU Measurements. *arXiv preprint arXiv:1610.01107* (2016).
[11] W. Knowles, D. Prince, D. Hutchison, J. Disso, and K. Jones. 2015. A survey of cyber security management in industrial control systems. *International journal of critical infrastructure protection* 9 (2015), 52–80.
[12] D. Kuipers. 2008. *Common Cyber Security Vulnerabilities Observed in Control System Assessments by the INL NSTB Program.* Technical Report. Idaho National Lab.(INL), Idaho Falls, ID (United States).
[13] J. Lamp, C. E. Rubio-Medrano, Z. Zhao, and G-J. Ahn. 2017. OntoEDS: Protecting Energy Delivery Systems by Collaboratively Analyzing Security Requirements. In *Collaboration and Internet Computing (CIC), 3rd Int. Conf. on.* IEEE, 1–10.
[14] R. M Lee, M. J. Assante, and T. Conway. 2016. Analysis of the Cyber Attack on the Ukrainian Power Grid. *SANS ICS Report* (2016).
[15] Seok Won Lee and Robin A Gandhi. 2005. Ontology-based active requirements engineering framework. In *Software Engineering Conference, 2005. APSEC'05. 12th Asia-Pacific.* IEEE, 8–pp.
[16] Seok-Won Lee, Robin A Gandhi, and Gail-Joon Ahn. 2007. Certification process artifacts defined as measurable units for software assurance. *Software Process: Improvement and Practice* 12, 2 (2007), 165–189.
[17] D. Lekkas and D. Spinellis. 2005. Handling and reporting security advisories: A scorecard approach. *IEEE Sec. & Priv.* 3, 4 (2005), 32–41.
[18] NERC. 2017. CIP Standards. www.nerc.com/pa/Stand/Pages/CIPStandards.aspx
[19] S. Peisert, R. Gentz, J. Boverhof, C. McParland, S. Engle, A. Elbashandy, and D. Gunter. 2017. LBNL Open Power Data. (2017).
[20] Smart Grid Interoperability Panel Cyber Security Working Group. 2010. Introduction to NISTIR 7628 guidelines for smart grid cyber security. *NIST Special Publication* 154 (2010).
[21] K Stouffer, S Lightman, V Pillitteri, M Abrams, and A Hahn. 2014. NIST Special Publication 800-82 Revision 2. *Gaithersburg, MD, USA: National Institute of Standards and Technology* (2014).
[22] US Department of Energy. 2017. *Quadrennial Energy Review Transforming the Nation's Electricity System: The Second Installment of the QER.*
[23] R. B. Vaughn, R. Henning, and A. Siraj. 2003. Information assurance measures and metrics-state of practice and proposed taxonomy. In *Proc. of the 36th Annual Hawaii Int. Conf. on System Sciences,.* IEEE, 10–pp.
[24] H. Yang, L. Cheng, and M. C. Chuah. 2018. Detecting Payload Attacks on Programmable Logic Controllers (PLCs). *2018 IEEE Conference on Communications and Network Security (CNS)* (2018), 1–9.