ExSol: Collaboratively Assessing Cybersecurity Risks for Protecting Energy Delivery Systems

Josephine Lamp*, Carlos E. Rubio-Medrano†, Ziming Zhao‡ and Gail-Joon Ahn†§
*University of Virginia, †Arizona State University, ‡Rochester Institute of Technology, §Samsung Research
Email: jl4rj@virginia.edu, crubiome@asu.edu, zxzics@rit.edu, gahn@asu.edu

***Abstract*—No longer just prophesized about, cyber-attacks to Energy Delivery Systems (EDS), e.g., the power grid, gas and oil industries, are now very real dangers, resulting in non-trivial economical losses and an erosion of the public's confidence in EDS infrastructures. In such a context, performing risk analysis for EDS is difficult due to their innate diversity and interdependencies, along with an always-increasing threatscape and attack vectors. With this in mind, this paper presents *ExSol*, a collaborative, real-time, requirements-based risk assessment framework that features an approach for modeling real-life EDS infrastructures, a technique that retrieves well-defined security requirements from an EDS ontology, and a methodology for calculating risk for a single asset and for an entire system. In addition, we also provide experimental evidence that includes several attack case scenarios, which showcase the effectiveness of our proposed approach for being fully deployed in practice.**

I. 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 [1]. Nowadays, these systems contain a high degree of automation used to efficiently manage their energy processes. Unsurprisingly, since EDS are critical components of a country's economy, they are high caliber targets for cyber-attackers. Attacks targeting EDS are no longer just prophesized about and instead are now a real danger; multiple attacks to EDS have occurred over the past 4 years in Ukraine, such as the Ukrenergo Attack (2016) [2].

Risk analysis has been proposed as a valuable way to protect EDS from such attacks as it identifies vulnerabilities and quantifies the impact of threats in order to develop more secure systems. For example, the *Roadmap to Achieve Energy Delivery Systems Cybersecurity* published by the Energy Sector Control Systems Working Group [3], contains a dedicated section about risk analysis and assessment, explaining the need for methodologies that evaluate system state and quantify 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, and there is a lot of room for missed vulnerabilities, making the development of accurate risk assessment measures to quantify such vulnerabilities a seemingly indomitable task [1].

In order to solve these problems, 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 quantify 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, allowing for a better understanding of system state, current threats, attack vectors and vulnerabilities, thus enabling the identification of zero-day vulnerabilities and the accurate quantification of system risk.

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 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.

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

II. Background & Related Work

Risk Definition and Metrics. Risk can be defined as the probability that a threat will exploit a particular vulnerability of a system [4]. We expand this definition to include the impact of security requirements: 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 quantified using security metrics at a variety of granularities throughout a system. For example, in the work of Lekkas and Diomidis [5], *vulnerability scorecards* full of metrics that quantify good and bad aspects of a system through a goal-question-metric technique are used to understand system risk.

Risk Assessment and Analysis. As mentioned in Section I, risk analysis has been proposed as a valuable way to evaluate and mitigate the potential risks within EDS. Capodieci et al. developed the MICIE project [6], 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. Additionally, in the CockpitCI project, developed by Cruz et al. [7], a detection system monitors and detects live threats within an EDS system and then uses this information to model risk using processing modules. MICIE uses a system-modeling approach, and relies on the model to identify unexpected events within the system based on previous measurements, and CockpitCI relies on intrusion detection mechanisms to identify potential threats. As a result, these approaches may not pick up all threats within a system, i.e., zero-day vulnerabilities, if the model fails to recognize new measurements it may not have yet seen (and therefore cannot monitor for.)

III. *ExSol*: Assessing Cybersecurity Risks for EDS

As mentioned in Section I, a way to accurately quantify risk within EDS, identify and address zero-day 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 quantify the amount of risk there is to an EDS. *ExSol*, as the name implies, 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.

A. System Modeling

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 quantification. For such a purpose, we relied on the observation of three key characteristics of EDS instances: First, within real 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, real EDS instances in the field, although heterogeneous and component diverse, generally have similar architectures.

Modeling Security Requirements. In order to tackle the security issues described in Section I, reputable organizations within the EDS community have published a series of documents detailing EDS architecture, security requirements and best practices. For example, the NIST 800-82 document [8] 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 [9], 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.

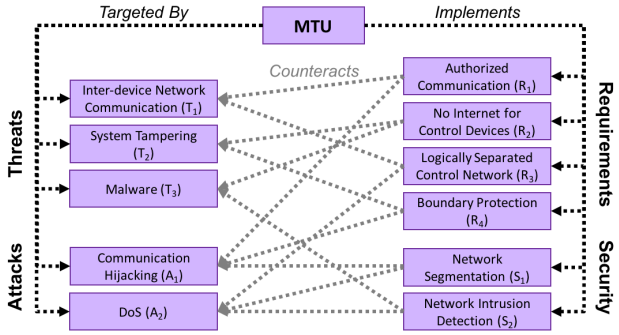

Figure 1: Example Risk Projection featuring the Threats (T_n), Attacks (A_n), Requirements (R_n), and Security Techniques (S_n) applicable to an MTU asset.

Risk Projections. In addition, for the purposes of risk assessment, information about security requirements and the functional relationships of EDS is crucial to properly quantify the risk to an asset based on their potential attacks, threats, and security solutions. With this in mind, we customized the aforementioned *OntoEDS* tool to develop our so-called *Risk Projections*, which pull out related requirements for an asset that are useful for risk quantification, e.g., 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. 1. 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. 1, 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 quad-tuple: <T, A, R, S>. An asset may have many quad-tuples, a.k.a., *Asset Objects*, based on the different combinations of T, A, R and S related to an asset.

System Templates. By using the Risk Projections just described, we develop abstract model representations, i.e., System Templates, for the EDS instances we wish to quantify 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). These templates contain a predetermined view of the system and their functional relationships, but can be expanded or changed to match exactly what a real infrastructure may look like. For example, a real system modeled by the System Template shown in Fig. 2 may include the assets Programmable Logic Controllers (PLCs), Master Terminal Units (MTUs), Controllers and a Human Machine Interface (HMI). Also, PLC_1 has a link to MTU_1 (because it

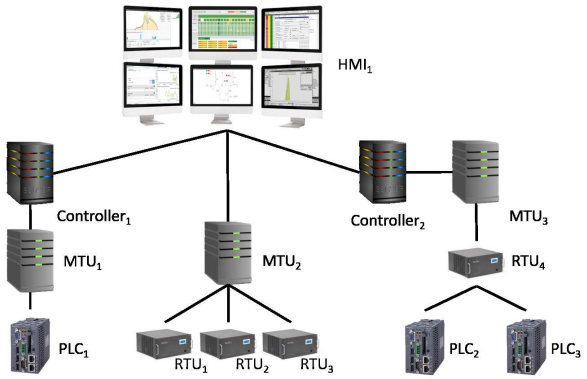

Figure 2: A System Template depicting our running example, developed using Risk Projections based on NIST 800-82 [8].

communicates with the MTU). Additionally, MTU_1 has a link to $Controller_1$ and $Controller_1$ has a link to HMI_1.

B. System Dependencies

Dependency Definition. In addition to modeling how EDS instances are architecturally set up, our System Templates also contain information about the 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. There are two main *types* of dependency relationships we define within *ExSol*: Command and Data Dependencies, which originate from the requirements summarized within our Risk Projections: either they send/receive *commands* or they send/receive *data*.

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 node and *to* the child node, e.g., parent→child. Functional Command Dependencies indicate that the parent asset sends *commands* to the child component, and Functional Data Dependencies indicate that the parent asset sends *data* to the child asset.

Security Dependencies. We define a security-related dependency as the following: *if an asset* As_1 *is compromised, and another asset* As_2 *is dependent on* As_1, *then* As_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 (As_1) security is impacted by a threat or attack, then the dependent child component (As_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. 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). 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.

Table I: 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

Dependency Matrix. Leveraging the definitions just presented, and in order to quantify the impact of dependencies between EDS assets, we developed a *Dependency Matrix* that allows the security relationships between all assets within a system to be quantified by specific factors. As described later in Section III-D, the matrix is useful in system *ExSol* calculation, where factors are applied to assets based on their dependencies before combining asset scores and calculating the final system score. This matrix is to be established collaboratively by the EDS community, using our Risk Projections as guidance about the dependencies between assets.

Example Matrix. An example Dependency Matrix, based on our System Template as shown in Fig. 2, is shown in Table I. 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 quantify the relationships between EDS system components.

C. ExSol *Calculation for a Single Asset*

Sub-Metrics for Exploitation Scores. In our *ExSol* approach, 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*. These metrics all quantify different aspects of the threat or attack, allowing for the accurate and comprehensive quantification of the Exploitation scores. A more detailed description of each of them is shown in Table II.

Sub-Metrics for Solution Scores. Conversely, solution scores within *ExSol* are composed of *Effectiveness*, *Relevance* and *Implementation* sub-metrics that are in turn intended to quantify the Requirements (*R*) and Security (*S*) for an asset. As with the metrics for Exploitation scores, a description is shown in Table II. Example metrics for an MTU are shown in Tables III and IV. 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 I. However, implementations of our *ExSol* approach can define their own scales based on their own customization needs.

Table II: *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*

Table III: Example ExSol T/A Metrics for an MTU

	T_1	T_2	T_3	A_1	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 IV: Example ExSol R/S Metrics for an MTU

	R_1	R_2	R_3	R_4	S_1	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**

Assigning Scores to Sub-metrics. In our *ExSol* approach, the aforementioned sub-metrics are developed in a collaborative fashion by types of experts in EDS infrastructures, denoted as Global and Local Experts. 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). Alternately, Local Experts *individually* determine the Local scores that quantify 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.

ExSol Asset Calculation Algorithm. With all this in mind, we now present how to calculate the actual *ExSol* scores for individual assets. The steps to calculate *ExSol* for an asset, along with an example for an MTU using Tables III and IV, are as follows:

Step 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 III, IV, and Fig. 1, 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).

Step 2. The Risk Projection next creates quad-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).

Step 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, 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 Table III and IV.

Step 4. Next, for each quad-tuple, the Exploitation sub-scores for each T and A, and the Solution sub-scores for each R and S are calculated. Equation 1 (1) and (2) are used for Threat and Attack metrics respectively and Equation 2 (3) and (4) are used for Requirement and Security metrics respectively.

Definition 1. *Exploitation Sub-score Calculations:*

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

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

where $\mathrm{T_i}$, $\mathrm{T_s}$, $\mathrm{T_r}$, $\mathrm{A_i}$, $\mathrm{A_s}$, and $\mathrm{A_r}$ stand as defined in Table II.

Definition 2. *Solution Sub-score Calculations:*

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

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

where $\mathrm{R_e}$, $\mathrm{R_r}$, $\mathrm{R_i}$, $\mathrm{S_e}$, $\mathrm{S_r}$, and $\mathrm{S_i}$ stand as defined in Table II.

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

Step 5. After the Exploitation and Solution sub-scores are calculated, the final *ExSol* score is calculated using Equation 3 (5) for each tuple.

Definition 3. ExSol *Score Calculation:*

$$ExSol = (R \cdot S) - (T \cdot A) \quad (5)$$

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* = ($R_1 \cdot S_1$) - ($T_1 \cdot A_1$) = (100 · 80) - (80 · 100) = 0.

D. ExSol *System-Wide Approach*

The process to calculate *ExSol* system-level risk scores, graphically displayed in Fig. 3, goes as follows:

Step 1. First, we obtain the set of assets included in the actual EDS infrastructure being evaluated. In our running example, such assets are the ones listed in Fig. 2.

Step 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 Fig. 2 and the Dependency Matrix in Table I are developed, and are also shown in Fig. 3 (1).

Step 3. Then, *ExSol* scores are calculated individually for each asset in the system, following the steps explained previously in Section III-C. 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 we calculate MTU_1's average *ExSol* as -613.

Step 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 quantification of the security dependencies between assets as determined using the Dependency Matrix, a percentage of the parent node's score is added to the child node's. Generally, this *percentage* dependency weight, denoted at %DW, can be calculated as:

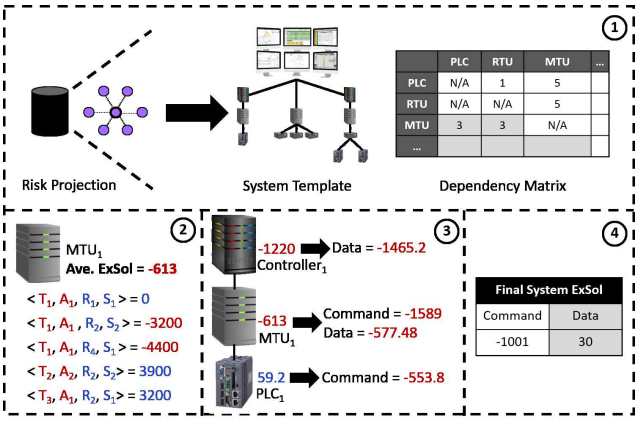

Figure 3: *ExSol* System Calculation Steps.

Definition 4. *Percentage Dependency Weight.*

$$\%DW_i = (DW_i \cdot 100)/MAX - SCALE \quad (6)$$

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 quantify the Dependency Matrix values, the dependency relationships between assets are properly quantified within the system *ExSol* score. The equation is shown next.

Definition 5. *Security Dependency Adaptions.*

$$Adapted\_\text{ExSol}_i = CN_i + (\%DW_i \cdot PN_i) \quad (7)$$

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. 3 (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.

Step 5. Finally, the system-wide *ExSol* score is calculated by choosing the lowest *Adapted_ExSol* score from all the assets within the EDS instance under study. 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: 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 quantify the risk for data-focused attacks, such as a Ladder Logic attack. In our example in Fig. 3 (4), we come up with the

following scores: Command System *ExSol* score = -1001 and Data System *ExSol* score = 30.

IV. Experimental Evaluation

Our experimentation was intended to test how *ExSol* reacted in realistic attack case scenarios where both Global and Local metrics change. With that in mind, we developed a variety of real-world attack scenarios based on common vulnerabilities as detailed by the U.S. Department of Energy [10], allowing us to observe 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 our experiments, we developed an EDS simulation infrastructure based on the System Template as shown in Figure 2, which uses a 48-core server running a combination of Virtualbox, OpenStack and Mininet, allocating multiple virtual machines (VMs) to simulate all relevant assets. Due to space constraints, we only elucidate a command attack scenario next.

Command Injection and Data Spoofing. Assuming the system shown in Fig. 2, 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 V (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 V (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 V (Part 3) are given.

Table V: Command Injection & Data Spoofing.

	Component	Original	Multiple	Percent
Part 1	$Controller_1$	0	-2770	-554
	MTU_1	0	-4155	-831
	MTU_2	0	-1587	-317.4
	RTU_2	-529	-529	-529
	PLC_1	-1385	-1385	-1385
Part 2	$Controller_1$	0	-2770	-554
	MTU_1	-140	-4295	-971
	MTU_2	-534	-2121	-851.4
	RTU_2	-529	-529	-529
	PLC_1	-1385	-1385	-1385
Part 3	$Controller_1$	0	642	128.4
	MTU_1	658	1578	807.6
	MTU_2	221	221	221
	RTU_2	837	837	837
	PLC_1	321	321	321

V. 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 quantify the amount of risk there is to an EDS, allowing for collaborators to work together and make group decisions about protection and mitigation actions for their infrastructures. In terms of future work, we plan on developing an optimization process that finds the *best* tuple pairs of security requirements for assets, potentially increasing the chances of *ExSol* for being adopted in practice.

Acknowledgments and Disclaimer

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. The work of Josephine Lamp and Ziming Zhao was performed during the time they were affiliated to Arizona State University. Any opinions, findings or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of United States Government or any agency thereof.

References

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

[2] Dragos Inc. (2017) CRASHOVERRIDE Analysis of the Threat to Electric Grid Operations. [Online]. Available: https://dragos.com/blog/crashoverride/CrashOverride-01.pdf

[3] E. S. C. S. W. Group *et al.*, "Roadmap to achieve energy delivery systems cybersecurity," *Energetics, Inc, URL https://www. controlsystemsroadmap. net/ieRoadmap\% 20Documents/roadmap. pdf*, 2011.

[4] R. B. Vaughn, R. Henning, and A. Siraj, "Information assurance measures and metrics-state of practice and proposed taxonomy," in *Proc. of the Hawaii Int. Conf. on System Sciences*,. IEEE, 2003, pp. 10–pp.

[5] D. Lekkas and D. Spinellis, "Handling and reporting security advisories: A scorecard approach," *IEEE Sec. & Priv.*, vol. 3, no. 4, pp. 32–41, 2005.

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

[7] T. Cruz, J. Proença, P. Simões, M. Aubigny, M. Ouedraogo, A. Graziano, and L. Yasakhetu, "Improving cyber-security awareness on industrial control systems: The cockpitci approach," in *13th Euro Conf. on Cyber Warfare and Sec.*, 2014, p. 59.

[8] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn, "Nist special publication 800-82 revision 2," *Gaithersburg, MD, USA: National Institute of Standards and Technology*, 2014.

[9] J. Lamp, C. E. Rubio-Medrano, Z. Zhao, and G.-J. Ahn, "Ontoeds: Protecting energy delivery systems by collaboratively analyzing security requirements," in *Collaboration and Internet Computing (CIC), 3rd Int. Conf. on*. IEEE, 2017, pp. 1–10.

[10] D. Kuipers, "Common cyber security vulnerabilities observed in control system assessments by the inl nstb program," Idaho National Lab.(INL), Idaho Falls, ID (United States), Tech. Rep., 2008.