# Model-Based Cybersecurity Assessment with NESCOR Failure Scenarios

Sumeet Jauhar*, Binbin Chen*, William G. Temple*, Xinshu Dong*,
Zbigniew Kalbarczyk†, William H. Sanders†, David M. Nicol†
*Advanced Digital Sciences Center, Illinois at Singapore, Singapore
{sumeet.j, binbin.chen, william.t, xinshu.dong}@adsc.com.sg
†University of Illinois at Urbana-Champaign, IL, USA
{kalbarcz, whs, dmnicol}@illinois.edu

**Abstract**

The transformation of traditional power systems to smart grids brings significant benefits, but also exposes the grids to various cyber threats. The recent effort led by US National Electric Sector Cybersecurity Organization Resource (NESCOR) Technical Working Group 1 to compile failure scenarios is an important initiative to document typical cybersecurity threats to smart grids. While these scenarios are an invaluable thought-aid, companies still face challenges in systematically and efficiently applying the failure scenarios to assess security risks for their specific infrastructure. In this work, we develop a model-based process for assessing the security risks from NESCOR failure scenarios. We extend our cybersecurity assessment tool, CyberSAGE, to support this process, and use it to analyze 25 failure scenarios. Our results show that CyberSAGE can generate precise and structured security argument graphs to quantitatively reason about the risk of each failure scenario. Further, CyberSAGE can significantly reduce the assessment effort by allowing the reuse of models across different failure scenarios, systems, and attacker profiles to perform "what if?" analysis.

## I. Introduction

Modernized electric power grids or smart grids incorporate information and communication technologies for improving power system control, monitoring, and response. However, the rapid digitization also exposes smart grids to various cyber threats. A recent report from the US Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) shows that cyber attacks targeting the energy sector dominate all such incidents with industrial control systems [1]. Neutralizing cyber threats in smart grids has become an urgent task, both for utility companies testing and deploying security mechanisms in their systems, and for the research community developing advanced solutions to combat emerging cyber threats. As part of this process, (cyber)security and risk assessment have become essential to support secure system design and deployment [2], [3], [4].

The electric sector failure scenarios and impact analyses, compiled by the US National Electric Sector Cybersecurity Organization Resource (NESCOR) Technical Working Group 1 [5], are a prominent example of the push toward more comprehensive and rigorous security assessment in the industry. The NESCOR failure scenarios describe specific types of undesirable cyber incidents and their impacts, as well as the vulnerabilities and potential mitigations associated with the failures. These scenarios, created by a coordinated community effort from grid operators, security consultants and regulators, provide an invaluable thought-aid for utilities to map to their systems, and identify relevant or similar threats to be addressed. However, companies still face challenges to systematically and efficiently apply these failure scenarios to assess the security risks of their specific infrastructure setups. In particular, following a manual thought process like this could fail to achieve its end goal due to the lack of accuracy, structure, convenience, and scalability.

Model-based security assessment methods can potentially be used to streamline this process. A significant amount of research effort has been devoted to formalize the description of systems, attackers, and other security-related information, and to automate the assessment processes as much as possible. However, there are a number of practical challenges in model-based security assessment, including *who creates the model?* and *how well does the model reflect the real system?* In the smart grid setting, we believe the NESCOR failure scenarios provide a promising foundation for model-based security assessment, because they describe realistic incidents that are of concern to the industry, and do so in a sufficient level of detail to allow model creation. However, to the best of our knowledge, no model-based security assessment tool exists today that can support the process of assessing a NESCOR failure scenario on a specific smart grid system.

In this work, we address this gap by integrating the NESCOR failure scenarios into a model-based security assessment process. We achieve this by extending and adapting the CyberSecurity Argument Graph Evaluation (CyberSAGE) software tool, which was developed in our previous research [6]. The goal of this work is two-fold: 1) to study the feasibility of applying a model-based approach to assess realistic threat scenarios in smart grids; 2) to demonstrate the benefits of having more structured, formalized, and mechanized approaches for assessing the security of smart grids.

To this end, we have studied one class of NESCOR failure scenarios (Distributed Energy Resources), and modeled them using an extended version of the formalism proposed in [7]. More specifically, we model each failure scenario using a mal-activity diagram similar to those proposed in [8], and formally represent the relevant vulnerabilities, mitigations, and attacker properties. Further, we capture the relationships between each type of information. The result of the security assessment is a *security argument graph* — a graphical representation that connects mal-activity process with relevant system components and threat agents. The *security argument graph* is generated automatically by our CyberSAGE tool based on the mechanisms introduced in [7]. The graph is also used to provide a quantitative risk assessment for the NESCOR scenario based on user-provided system and attacker properties. Our model-based assessment process and the accompanying CyberSAGE tool makes it effective to perform "what if?" analysis for varying system settings and attacker profiles. Furthermore, CyberSAGE can significantly reduce the analyst's assessment effort by allowing the reuse of models across different failure scenarios, systems, and attacker profiles.

## II. Background

Cybersecurity for smart grid systems having become a major concern over the last few years, it has become essential for utility companies to understand the latest threats and conduct thorough risk assessment for their systems. In this section, we introduce the NESCOR failure scenarios for unfamiliar readers and discuss model-based security assessment, to provide a foundation for the subsequent discussion of model-based cybersecurity assessment with the failure scenarios.

### A. NESCOR Failure Scenarios

NESCOR Technical Working Group 1 has spent years developing and refining a set of electric sector failure scenarios and impact analyses [5]. The NESCOR failure scenarios consist of 111 unique cyber-incidents that could negatively impact an electric utility. Those scenarios cover a number of smart grid domains:

- Advanced Metering Infrastructure (32 scenarios)
- Distributed Energy Resources (25)
- Wide-Area Monitoring, Protection, and Control (11)
- Electric Transportation (16)
- Demand Response (7)
- Distribution Grid Management (16)
- Generic (4)

Each NESCOR scenario consists of a written **description**, a list of **relevant vulnerabilities**, a list of **impacts**, and a list of **potential mitigations** [9] (see example in Section III). The scenarios are intended to help utility companies conduct risk assessment [10], as well as to improve cybersecurity awareness during procurement/planning and to assist with training. They have also drawn research interest [11], [12], [13].

While the types of failures considered in the NESCOR scenarios are far-reaching and clearly defined, the process of analyzing a specific system for a scenario-based risk assessment is effort intensive and largely subjective. New tools are needed to help utilities maximize the benefits of NESCOR's efforts. Such tools would reduce human effort, promote repeatability of results, and clarify the assumptions and information that serve as the starting point for an assessment. For those tasks, the field of model-based security assessment has much to offer.

### B. Model-Based Cybersecurity Assessment

Over the years, the cybersecurity research community has been working to develop various model-based tools to support security and risk assessment. The model-based tools available today include UML-based approaches [14],

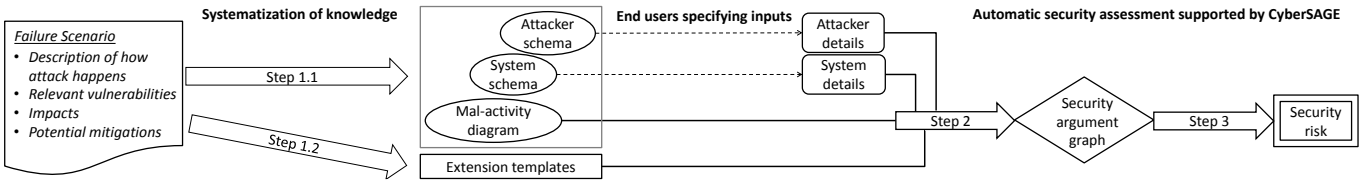


Fig. 1: The main steps of our approach in conducting model-based security assessment with NESCOR failure scenarios.

petri-net-based approaches [15], attack-tree-based approaches [16], and hybrid approaches combining different inputs [6]. Regardless of the modeling formalism, such tools share many common objectives and features. For example, many support a quantitative evaluation of system security using various metrics. The stated aim is often to support decision making and various comparative analyses.

In this paper, we focus on the CyberSecurity Argument Graph Evaluation (CyberSAGE) approach and software tool. This methodology uses workflow models to define the scope of the security assessment: for example, allowing a utility to assess the security of its smart meter reading processes [17], or a distribution grid measurement and control process [7]. The assessment connects different types of information, including:

- **Goal:** a system-level property or requirement for the specified workflow (e.g. availability).
- **Workflow:** a model of the actors and interactions occurring in the system (e.g., UML activity diagram).
- **System:** a model describing the system's devices, connections, and configurations.
- **Attacker:** a model describing the skills, resources, and knowledge of the attacker under consideration.

Each of these elements are combined to form a *security argument graph* that visually represents potential attacks on the system components implementing a workflow. The structure of this graph determines dependency relationships that can be used to calculate system-level metrics from low-level data. The graph itself is created automatically, provided the above inputs are present. This is done through a library of *extension templates* that determine how various pieces of information are connected. For example, there could be a rule connecting a workflow step by a certain type of actor (e.g., intelligent electronic device) to a specific device in the network (e.g., EV charging station). More details may be found in our previous work [7]. To model the NESCOR failure scenarios in this paper, new extension templates are needed, and these are discussed in the next section.

### C. Other Approaches for Cybersecurity Assessment

There are a number of standards and guidelines that are relevant to security and risk assessment for smart grid systems. The National Institute of Standards and Technology (NIST) has released a document that provides comprehensive guidelines on the requirements for and properties of secure smart grid systems [4]. It also highlights the importance and desirable goals of security and risk assessment. Similarly, the North American Electric Reliability Corporation (NERC) guidelines elaborate further on detailed aspects that a security assessment needs to cover for smart grid [18]. Serving as a more general guide beyond just smart grid as the application, NIST Special Publication 800-53 presents an exhaustive list of *security and privacy controls* that can be leveraged to compose the dimensions of the system to be assessed [19].

A number of industry standards and best practices, including those from NIST and NERC, may be assessed using the Cyber Security Evaluation Tool (CSET) [20] from ICS-CERT. This questionnaire-based tool helps with the generation of compliance documentation and helps to couple the general standards with a user-specified network diagram. Finally, more relevant to this paper, EPRI has developed a Microsoft Excel toolkit to support the evaluation of NESCOR failure scenarios [21]. We discuss the merits of our approach compared to the EPRI toolkit as part of our evaluation in Section IV.

## III. From Scenarios to Model-based Assessments

While the NESCOR scenarios are an invaluable thought-aid and education tool, we see the potential benefits of systematizing the knowledge embedded in the NESCOR documents and using them as a basis to conduct model-based assessments. We develop an extension of our workflow-oriented security assessment approach [7], [17] to realize this vision. This section describes the steps and intuition behind our method.

### A. Running Example: NESCOR Scenario DER.1

We use the first distributed energy resource failure scenario (DER.1), *Inadequate Access Control of DER Systems causes Electrocution*), throughout the rest of this paper to illustrate our model-based security assessment method and results. We reproduce DER.1 below from [5] in slightly abridged form for readers' easy reference.

**Description:** *The DER owner fails to change the default password or not set a password for the DER system user interface. A threat agent (inept installer, hacker, or industrial spy) gets access through the user interface and changes the DER settings so that it does not trip off upon low voltage (anti-islanding protection), but continues to provide power during a power system fault.*

**Relevant Vulnerabilities:**

- Physical access may be obtained by unauthorized individuals to DER settings through the system UI
- Default password is not changed for the DER system
- System permits unauthorized changes to anti-islanding protection due to poor configuration design
- Commands or other messages may be inserted on the network between the user interface and the DER system, that result in unauthenticated changes to sensitive parameters

**Impacts:**

- System suffers physical damage due to feeding into a fault
- A utility field crew member may be electrocuted
- The utility experiences damage to its reputation

**Potential Mitigations:**

- Authenticate users for all user interface interactions
- Change default access credentials after installation
- Enforce limits in hardware to prevent equipment damage
- Train personnel on secure networking requirements
- Require management approval for critical security settings

### B. Model-Based Security Assessment Process

Our approach models the information captured in NESCOR failure scenarios to assess the security level of a specified system against a specific attacker. Fig. 1 summarizes the main steps of our approach. We start by the systematization of the knowledge embedded in the failure scenarios into individual models (step 1.1) and their logical relationships (step 1.2). An end user then can specify the details about her system and the concerned attacker, based on which the CyberSAGE tool supports the automatic generation of a security argument graph (step 2) and quantitative evaluation of the security risk (step 3). We now discuss each step in more details using the DER.1 example.

**Step 1: Systematization of knowledge**: This step consists of two sub-steps as described below.

**Step 1.1 (Representation of information)**: This sub-step formalizes different aspects embedded in failure scenarios. Specifically, we represent the textual description of how a failure happens as a mal-activity diagram (similar to [8]). We use the vulnerabilities and mitigations to define schemas of system/attacker properties. As we will show in Section IV, many properties defined in our schemas (like security controls of a device) are applicable across multiple failure scenarios. We also associate the impacts with the mal-activity diagram. Our CyberSAGE tool supports the creation of mal-activity diagrams, and the customization of system and attacker schemas.

For the DER.1 example, we begin by developing a mal-activity diagram (see Fig. 2) to show how a threat agent's actions may influence the normal processes in the system. The threat agent first tries to gain either physical or network access to the DER-HMI, then changes the settings. This malicious activity results in the system taking an unsafe action. We also use the scenario description to identify key system components (e.g., DER-HMI and DER equipment). We associate each type of system with a schema of properties, which are derived based on the vulnerabilities and mitigations in the NESCOR document. Example properties include *restrict-physical-access*, *authenticate-user*, etc. We also define an attacker schema to capture attacker properties that affect the outcome of individual attack steps, such as *access*, *domain knowledge*, *IT skills* and *malicious intention*.

**Step 1.2 (Extension templates)**: After formalizing different aspects of information (mal-activity diagram, system, and attacker schemas), we must systematize the logical relationships among them. This is done using *extension templates* as we defined in our previous work [7]. Intuitively, an extension template is a formal reusable rule for

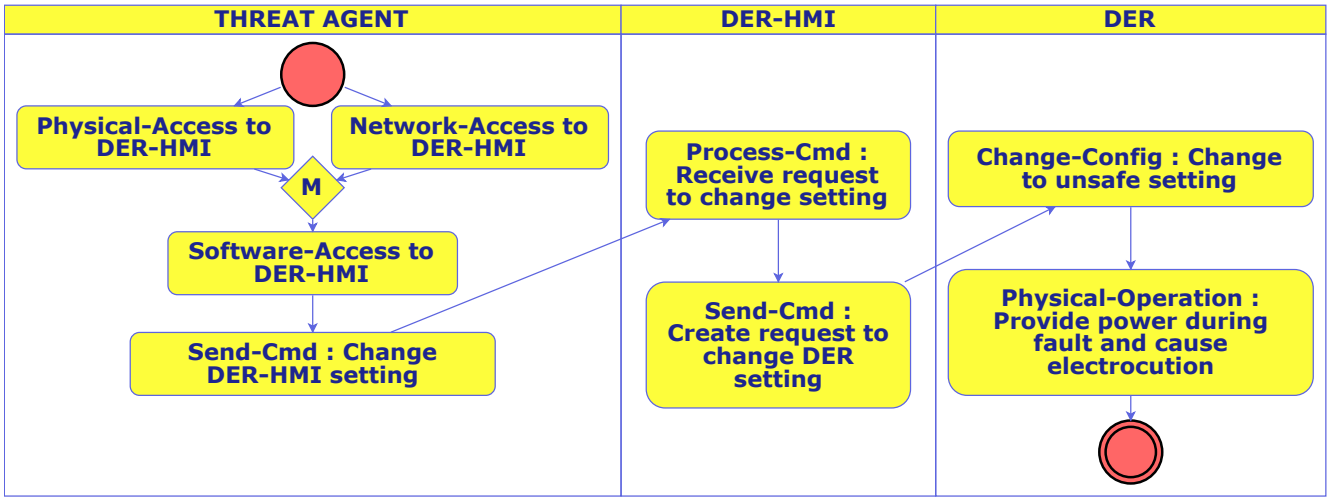


Fig. 2: DER.1 mal-activity diagram (CyberSAGE screenshot).

| Vulnerabilities | Attacker property | | $p$ |
|---|---|---|---|
| Weak authentication | IT skills | H | 0.80 |
| | | M | 0.4 |
| | | L | 0.1 |
| Unchanged default password | IT skills | H | 0.99 |
| | | M | 0.95 |
| | | L | 0.9 |

TABLE I: Probability table (excerpted) of *Software Access* extension template.

connecting a security-related statement or claim with relevant supporting arguments, which also carries the logic about how numerical evidence associated with supporting arguments affect metrics associated with the higher level statement. Our CyberSAGE tool supports the description of extension templates as Drools rule files[1].

We have studied all 25 DER examples, for which we look beyond the templates from our previous work [7] and define 15 new extension templates. For the DER.1 example, 7 out of these 15 are applicable: *Software Access*, *Network Access*, *Physical Access*, *Send Command*, *Process Command*, *Change Configuration* and *Physical Operation*.

As an example, the logic in the *Software Access* extension template specifies that *access to software on a specific device* depends on: 1) its previous activity step, i.e., access to the corresponding device (either physically or through network), and 2) the relevant system properties (e.g., the access control mechanisms for the particular software), and 3) the requirement for the attacker (e.g., her access to credential information and her IT skill level).

Our *Software Access* extension template also quantifies the probability of exploiting vulnerabilities to gain software access as a function of system and attacker properties. For example, Table I shows that an attacker can gain software access by exploiting either *weak authentication* or *unchanged default password*. It specifies the conditional probability, denoted by $p$, of successfully exploiting some vulnerability given a certain attacker property. The example values in the table show that an attacker with high IT skill can exploit the weak authentication with probability $p = 0.8$, while an attacker with low IT skill can only do so with probability of $p = 0.1$. On the other hand, exploiting the *unchanged default password* vulnerability is less sensitive to an attacker's IT skill level. In the example, $p \geq 0.9$ even for an attacker with low IT skill.

**Step 2: Security argument graph generation**: The first step distils the knowledge embedded in a failure scenario by breaking different pieces apart and identifying their logical relationships. Step 2 uses this information to examine a specific system, by creating a security argument graph that connects security-related information to reason about the security level of the studied system. As opposed to the examples in our earlier work [7], [17] where the security argument graph is to argue about a positive goal (e.g., system availability), in the NESCOR failure scenario analysis we use the threat agent's goal (i.e., to cause the failure to occur) as the focal point to generate the security argument graph. To generate the security argument graph, end users need to first specify the details about their system based on the schema from step 1.1, then the CyberSAGE tool can automatically apply suitable extension templates from

[1]http://www.drools.org

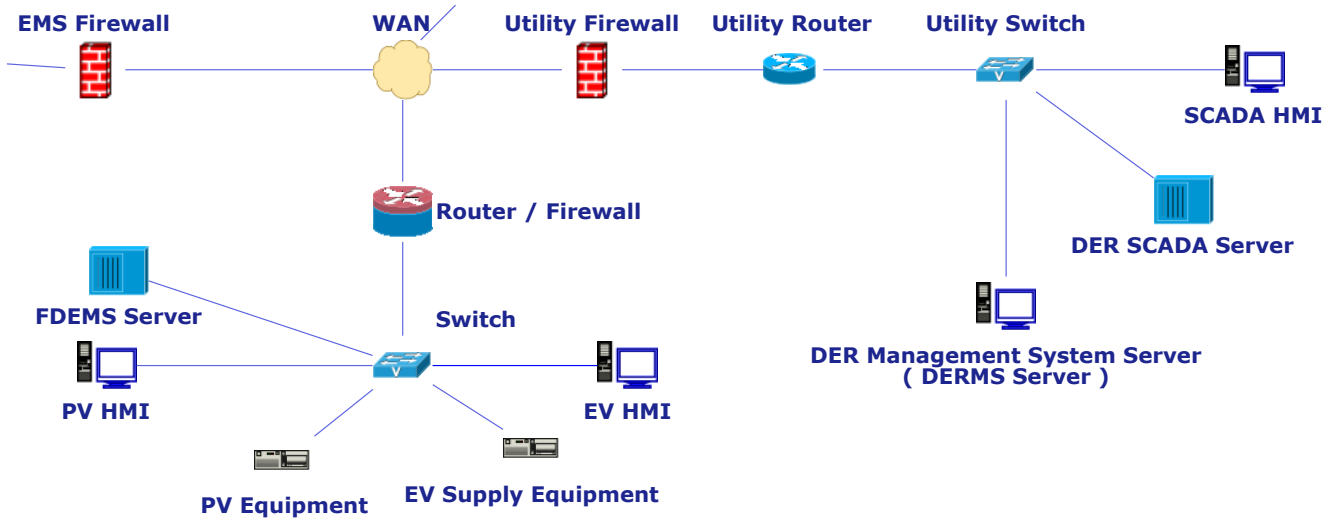


Fig. 3: A partial system topology (CyberSAGE screenshot), showing components relevant to DER.1.

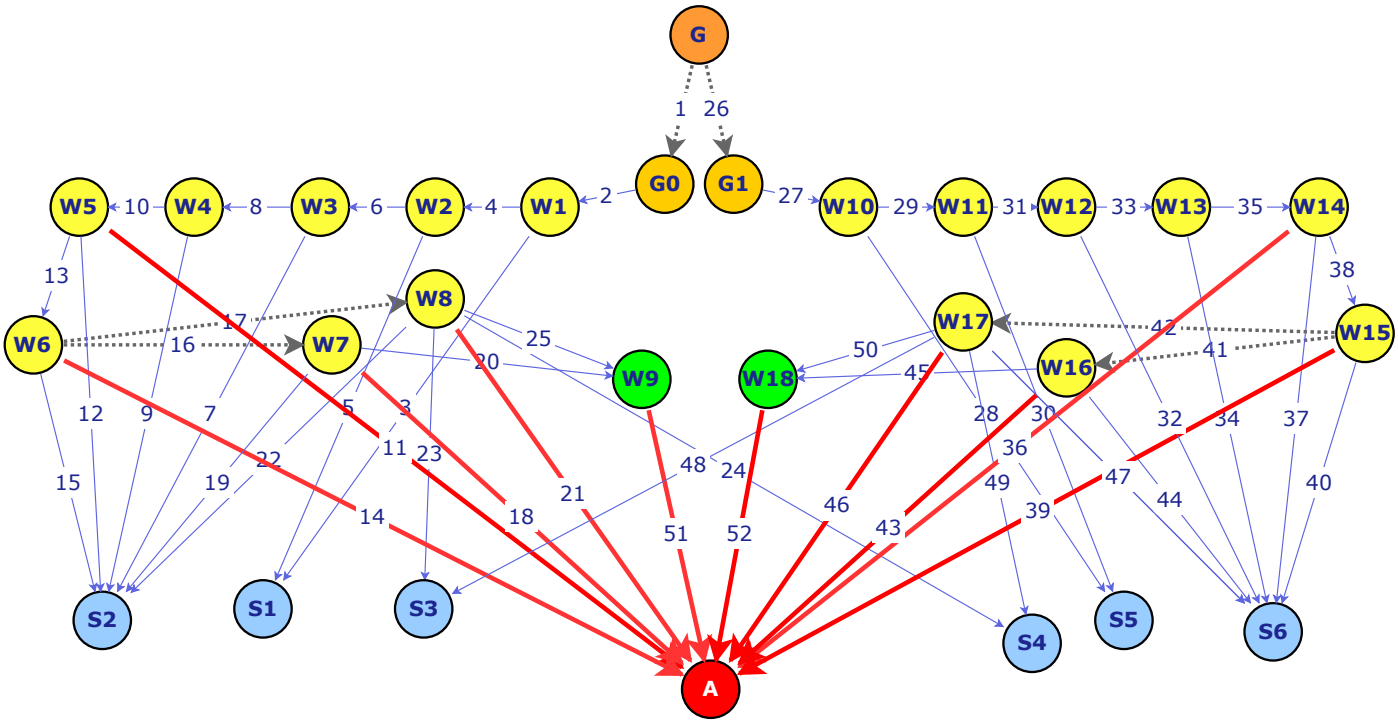


Fig. 4: DER.1 security argument graph generated for system topology in Fig. 3 (CyberSAGE screenshot).

step 1.2 in a recursive manner to integrate the mal-activity diagram, system, and attacker information.

To analyze the DER.1 failure scenario, an analyst needs to supply a system topology (see a simple example in Fig. 3) to CyberSAGE and indicates how the actors in the mal-activity diagram (see Fig. 2) map to the components in the system topology. For this example, we run a security assessment for a DER management system with a photovoltaic system and an electric vehicle charging station. The resulting security argument graph is shown in Fig 4. From top to bottom, the $G$ nodes represent the attacker's goal, the yellow $W$ nodes represent steps in the activity diagram (with the special green nodes indicating the starting state), the blue $S$ nodes represent the system components, and finally the $A$ node represents the attacker. As shown in Fig. 4, the goal node $G$ (the failure scenario occurs) holds if either $G0$ or $G1$ holds, which are two sub-goals that correspond to failure of the photovoltaic system and electric vehicle charging station respectively. For attack to accomplish goal node G0, its previous activity step (i.e., physical-operation at DER device) must occur (represented by node W1), which in turn depends on its previous activity step (i.e., Change-Config at DER, represented by node W2) and properties of the component corresponding

to its actor (i.e., the DER, represented by node S1). A security argument graph also visualizes the attacker's impact on the system. For example, all the thick red edges in the graph show how a particular mal-activity step relates to certain properties of an attacker.

**Step 3: Security metrics calculation**: The final step of our security assessment method uses the generated security argument graph to quantitatively evaluate the risk of the NESCOR failure scenario. The evaluation is based on the logical relationships among different nodes, as defined by the corresponding extension templates. Specifically, each mal-activity step in the graph will be associated with some probability that is computed based on both the relevant vulnerabilities and the concrete properties of the concerned attacker (see Table I for an example). CyberSAGE then aggregates the probability values of these individual events through the logical operators (AND / OR) to calculate the probability of the goal node in the security argument graph.

We have specified an example set of such relationships in our extension templates (see Appendix B). In Section IV we provide more details about DER.1 evaluation results based on our example extension templates. A security analyst can easily customize the evaluation logic by editing the corresponding extension templates.

## IV. Evaluation

This section reports the overhead (or the effort required) and benefits of applying our model-based assessment method with NESCOR failure scenarios. Our evaluation is based on an extended version of our CyberSAGE tool, which allows automatic generation of security argument graphs and automatic evaluation of security metrics like failure probabilities.

### *A. Effort Required to Apply a Model-based Approach*

With the support of our CyberSAGE tool, we have modeled all 25 NESCOR failure scenarios under the DER category. As discussed in Section III (see Fig. 1), applying our approach requires manual effort to model the mal-activity diagrams and system / attacker schemas (step 1.1), and to distil the extension templates (step 1.2). One researcher spent around 1 hour to formalize each of the 25 different mal-activity diagrams based on the description of each individual scenario. Also we distil the system components involved in these scenarios into 10 different types, including DER device, DER Human-Machine-Interface (HMI), server, networking devices, etc. We define for each type a corresponding schema of security related properties (e.g., access control, authentication, etc.), based on the vulnerabilities and potential mitigations mentioned in the scenarios. In total, less than 100 distinct properties are needed to model all devices mentioned in the 25 scenarios. Since the same system and attacker templates are shared across different scenarios, the modeling effort is effectively amortized.

After we had all 25 mal-activity diagrams in place, we conducted a 2-day internal workshop, identifying common attack steps shared by different scenarios and creating extension templates to describe them. We find that we only need 15 extension templates to cover all of the 25 DER failure scenarios. This is because many failure scenarios share similar types of attack steps (e.g., gaining network access, sending malicious commands). Furthermore, we find that most of the 15 extension templates we identified can be readily applied to failure scenarios in other NESCOR failure scenario categories, such as those related to Advanced Metering Infrastructure (AMI). Due to space limitations, we report the details of our 15 extension templates in Appendix B.

Once the above knowledge systematization step has been done, an end user only needs to specify the details of her concerned system and attackers based on the defined schema. By following our method, this becomes a one-time effort, since the user inputs can be reused across different DER failure scenarios. Finally, once the user specifies the inputs, CyberSAGE automates the remaining steps, i.e., the generation of the graph and the evaluation of the security metrics.

Compared to the CyberSAGE tool, the EPRI Excel toolkit [21] places greater reliance on the analyst's judgment when evaluating the risks from failure scenarios. The toolkit allows the analyst to go through each failure scenario individually and rate the severity of vulnerabilities (low/moderate/high), the implementation level of mitigations (not/partially/largely/fully implemented), and threat and impact scores (on a scale of 0,1,3,9). Based on the total number of likelihood and impact points, the analyst then manually assigns a risk score of high/moderate/low. Since there is no explicit data input from or modeling of the system being assessed, the time needed in the EPRI Excel toolkit for assessing an individual scenario is similar or potentially less than CyberSAGE. However, when scaling up to dozens or hundreds of scenarios, a CyberSAGE assessment can be less ambiguous, more consistent, and simultaneously more efficient, since it supports longer-term reduction in effort from prior systematization.

| | IT Skill | Domain knowledge | Access | Probability to launch (C / I / A) attack |
|---|---|---|---|---|
| Inept installer | Low | Low | Physical | 0.1 / 0.1 / 0.1 |
| Hacker | High | Medium | Remote | 0.6 / 0.9 / 0.9 |
| Industrial spy | High | High | Remote | 0.9 / 0.6 / 0.6 |

TABLE II: Main properties of three attacker profiles.

| | Inept installer | Hacker | Industrial spy |
|---|---|---|---|
| Baseline (no mitigation) | $2.1 \times 10^{-2}$ | $5.0 \times 10^{-1}$ | $5.3 \times 10^{-1}$ |
| Authenticate Users | $1.6 \times 10^{-2}$ | $3.8 \times 10^{-1}$ | $4.0 \times 10^{-1}$ |
| Change default password | $1.6 \times 10^{-2}$ | $3.9 \times 10^{-1}$ | $4.0 \times 10^{-1}$ |
| Enforce hardware limits | $4.2 \times 10^{-5}$ | $1.0 \times 10^{-3}$ | $1.0 \times 10^{-3}$ |
| Secure network training | $2.1 \times 10^{-2}$ | $1.5 \times 10^{-1}$ | $1.5 \times 10^{-1}$ |
| Require manager approval | $1.3 \times 10^{-2}$ | $3.1 \times 10^{-1}$ | $3.3 \times 10^{-1}$ |
| All mitigations | $6.4 \times 10^{-7}$ | $6.9 \times 10^{-7}$ | $7.3 \times 10^{-7}$ |

TABLE III: Probability for DER.1 failure scenario to occur.

### *B. Assessment under Varying System and Attacker Settings*

To provide a concrete example of the potential benefits from our approach, we describe how a security analyst can use CyberSAGE to study the probability for failure scenario DER.1 to occur under different system settings and different attacker profiles. We start with a baseline setting where no mitigations are in place, and then progressively apply different mitigations one by one, according to the suggestions in the NESCOR document for DER.1. We model three attacker profiles—inept installer, hacker, and industrial spy—as mentioned in the DER.1 scenario description. Table II summarizes the main properties of the three attacker profiles, including their level of IT skill and domain knowledge, their possible access levels to the DER system and their malicious intention, as described by the probability of launching attacks impacting confidentiality/integrity/availability. All the parameters specified in Table II can be fine-tuned by a security analyst based on her experience.

CyberSAGE supports quantitative evaluation of the probability for a failure scenario to occur. This can be extended to a risk score for the failure scenario by multiplying the failure probability by an impact score provided during model creation. A user can easily vary system settings and attacker properties in CyberSAGE. The assessment result depends on the structure of the security argument graph, the system and attacker inputs, and the probability tables embedded in the relevant extension templates. Due to space limitations, we only provide an intuitive interpretation of the results in this section. Appendix B contains more detail about the system settings, attacker properties, extension template coefficients, including a preliminary sensitivity analysis of the quantitative input values.

As shown in Table III, under the baseline setting, for both the industrial spy and hacker profiles, there is a high probability (around $0.5$) that the attacker can break into the system by exploiting network-related vulnerabilities, ultimately causing failure DER.1 to happen. Note that, although an industrial spy is more capable than a hacker (as she has better domain knowledge and similar IT skill), her intent to cause an integrity-related failure scenario like DER.1 is lower than a hacker's. These factors are aggregated through the security argument graph and result in similar failure probabilities under the two attacker profiles.

The table also shows that CyberSAGE can differentiate among the impacts of various mitigations for different types of attackers. For example, the mitigation of enforcing hardware limits on the DER device has a bigger impact on preventing DER.1 failure scenario against all attackers. In comparison, training the personnel on secure networking requirements can reduce the failure probability for both the hacker and industrial spy settings, but not for the inept installer setting. This is because the installer gains software access through physical means instead of through network. The last row of the table shows that applying all mitigations on the system results in an even lower chance that the failure scenario can occur. Since applying all mitigations can be expensive, an analyst can use CyberSAGE to study the failure probability under different subset of relevant mitigations to identify more effective mitigations.

## V. Conclusion

In this paper, we develop a model-based method for assessing the security risks of NESCOR failure scenarios. With the support from an extended version of our model-based security assessment tool CyberSAGE, we demonstrate the effectiveness of our proposed method. Incorporating realistic failure scenarios into model-based security assessment tools can aid security analysts to systematically and efficiently assess the security level of their systems. We will make our failure scenario models available through CyberSAGE tool portal [22]. We hope our work will facilitate the adoption of model-based security assessment in the smart grid industry.

## References


[1] US Department of Homeland Security, "ICS-CERT monitor," Feb. 2015.

[2] ENISA, "Appropriate security measures for smart grids," 2012.

[3] R. Habash, V. Groza, D. Krewski, and G. Paoli, "A risk assessment framework for the smart grid," in *IEEE EPEC*, Aug. 2013.

[4] NIST: The Smart Grid Interoperability Panel - Cyber Security Working Group, "NISTIR 7628, Guidelines for Smart Grid Cyber Security, Revision 1," 2014.

[5] National Electric Sector Cybersecurity Organization Resource, "Electric sector failure scenarios and impact analyses," Electric Power Research Institute, Tech. Rep. 2.0, Jun. 2014.

[6] A. H. Vu, N. O. Tippenhauer, B. Chen, D. M. Nicol, and Z. Kalbarczyk, "CyberSAGE: A tool for automatic security assessment of cyber-physical systems," in *QEST*, Sep. 2014.

[7] N. O. Tippenhauer, W. G. Temple, A. H. Vu, B. Chen, D. M. Nicol, Z. Kalbarczyk, and W. Sanders, "Automatic generation of security argument graphs," in *PRDC*, Nov. 2014.

[8] G. Sindre, "Mal-activity diagrams for capturing attacks on business processes," in *REFSQ*, Jun. 2007.

[9] National Electric Sector Cybersecurity Organization Resource, "Electric sector failure scenarios common vulnerabilities and mitigations mapping," Electric Power Research Institute, Tech. Rep., Jun. 2014.

[10] A. Lee, "Integrating electricity subsector failure scenarios into a risk assessment methodology," Electric Power Research Institute, Tech. Rep., Dec. 2013.

[11] R. Abercrombie, B. Schlicher, and F. Sheldon, "Security analysis of selected ami failure scenarios using agent based game theoretic simulation," in *HICSS*, Jan. 2014.

[12] R. Berthier and W. H. Sanders., "Monitoring advanced metering infrastructures with amilyzer," in *C&ESAR*, Nov. 2013.

[13] C. Hawk and A. Kaushiva, "Cybersecurity and the smarter grid," *The Electricity Journal*, vol. 27, no. 8, pp. 84 – 95, 2014.

[14] T. Sommestad, M. Ekstedt, and H. Holm, "The cyber security modeling language: A tool for assessing the vulnerability of enterprise system architectures," *Systems Journal, IEEE*, vol. 7, no. 3, Sep. 2013.

[15] E. LeMay, M. Ford, K. Keefe, W. Sanders, and C. Muehrke, "Model-based security metrics using ADversary VIew Security Evaluation (ADVISE)," in *QEST*, Sep. 2011.

[16] B. Kordy, L. Pietre-Cambacedes, and P. Schweitzer, "DAG-based attack and defense modeling: Don't miss the forest for the attack trees," *CoRR*, vol. abs/1303.7397, 2013.

[17] B. Chen, Z. Kalbarczyk, D. M. Nicol, W. H. Sanders, R. Tan, W. G. Temple, N. O. Tippenhauer, A. H. Vu, and D. K. Yau, "Go with the flow: Toward workflow-oriented security assessment," in *NSPW*, 2013.

[18] The North American Electric Reliability Corporation (NERC), "Security guidelines for the electricity sector: Vulnerability and risk assessment, version 1.0," 2012.

[19] NIST: Joint Task Force Transformation Initiative, "Nist special publication 800-53, security and privacy controls for federal information systems and organizations, revision 4," 2013.

[20] ICS-CERT, "CSET: The cyber security evaluation tool."

[21] EPRI, "Failure scenario based risk assessment toolkit for the electricity subsector," June 2014, prototype Version 0.3.

[22] "CyberSAGE," https://www.illinois.adsc.com.sg/cybersage/.

# APPENDIX A
# INPUT PREPARATION FOR CYBERSAGE

The model-based approach implemented in CyberSAGE starts with the systematization of the knowledge embedded in the textual descriptions of the NESCOR document, by converting them into mal-activity diagrams, system model schemas, and attacker model schemas. This section presents a detailed description of the various input models.

## A. Mal-activity Diagram Modeling

We develop mal-activity diagrams for all the 25 failure scenarios under the DER category, according to the text descriptions. The mal-activity diagrams focus on the modeling of the malicious activities of the attacker, but also include how the changes induced by the attackers propagate through the systems. The actors of the mal-activity diagram is related to attacker and different system components. The logical relationship between a activity step and the related system and attacker properties are expressed in the form of extension templates, which we will further discuss in Appendix B.

We classify each mal-activity diagram based on the kind of potential impact it would have on the smart grid. For example, scenario DER.1 will result in a compromise in the integrity of the smart grid. Other failure scenarios may result in a compromise of the availability, integrity, confidentially and/or a mix of all on the smart grid. We use this classification to decide the level of intention for a particular type of attacker to carry out the corresponding attack, i.e., an attacker's probability to launch a specific type of attack. Security analysts can also associate with each mal-activity diagram a quantitative value of impact. CyberSAGE allows the computation of the success probability of an attack based on the mal-activity diagram, and related system and attacker properties (see details of an example evaluation logic for computing the success probability in Appendix C). Combining all three, one can calculate the overall risk for the corresponding failure scenario as:

$$\text{Risk} = Prob[\text{launch an attack}] \times Prob[\text{Success of the attack}|\text{attack is launched}] \times \text{Impact}$$

## B. System Modeling

NESCOR defines a set of 85 vulnerabilities applicable to smart grid domain, which can be classified into 22 vulnerability classes according to NIST's cyber security guidelines for smart grid systems [4]. These vulnerability classes give useful insight into potential people, policy, operational, software/firmware and network related weaknesses in the system that can have adverse impacts on the correct operations of the smart grid. CyberSAGE's security argument graph generation engine associates a specific attack activity with a set of vulnerabilities. For example, in order to *gain software access*, one can exploit the *weak authentication* or *unchanged default password* vulnerabilities. For each vulnerability, we associate it with a list of potential mitigations according to NESCOR document. The NESCOR document provides a comprehensive set of 171 mitigations. CyberSAGE maps these mitigations to appropriate devices as their properties. CyberSAGE allows users to define the effectiveness of relevant mitigations in various devices. Table IV lists a set of devices that we infer from the specifications of the DER scenarios, as well as their properties (inferred from appropriate mitigations).

Table IV lists the devices and their properties, as derived from the NESCOR documents. Besides that, to model the complete system topology, we also need to define an additional set of devices as in Table V, which are mostly networking devices.

## C. Attacker Modeling

The NESCOR document describes a concrete set of threat agents that may compromise the operations of the smart grid. These include economic criminals, malicious criminals, recreational criminals, activists, terrorists and (non-malicious) hazards. It takes into account the motivation, tactics and capabilities of these threat agents to launch attacks on the smart grid. The attacker model in our current CyberSAGE implementation captures the capabilities, intention (motivation), and the access of potential threat agents.

- Skill Level: We consider the IT and domain skills needed by the threat agent to launch various types of attacks. For example, exploiting a software vulnerability to escalate privilege needs IT skills, while the sending legitimate

| Device | List of properties |
|---|---|
| DER HMI | Protect Credentials<br>Secure Factory Settings<br>Restrict Physical Access<br>Restrict Remote Access<br>Restrict Application Access<br>Enforce Restrictive Firewall Rules<br>Read Only Access For Critical Configuration<br>Require Approval For Critical Security Settings<br>Limit Remote Modification<br>Generate Security Alarms<br>Authenticate Devices<br>Authenticate Messages<br>Authenticate Users<br>Enforce Changing Default Passwords<br>Create Audit Logs<br>Protect Audit Logs<br>Security Training For Personal<br>Manager Approval For Critical Changes<br>Confirmation For Security Changes<br>Configured For Least Functionality |
| DER equipment | Enforce Limits in Hardware |
| FDEMS | Restrict Application Access<br>Enforce Least Privilege<br>Enforce Restrictive Firewall Rules<br>Authenticate Users<br>Enforce Changing Default Passwords<br>Require Multi Factor Authentication<br>Protect Credentials<br>Security Training For Personal<br>Application Whitelisting<br>Secure Boot Loader<br>Maintain Patches<br>Generate Alerts For Critical Events Or Operations<br>Check OS Integrity<br>Check Software Execution Integrity<br>Check Software File Integrity<br>Create Audit Logs |
| SCADA Server | Restrict Application Access<br>Confirm Before Microgrid Disconnection<br>Intrusion Detection And Prevention<br>Authenticate Devices<br>Authenticate Messages<br>Authenticate Users<br>Protect-Credentials |
| DERMS | Protect Credentials<br>Restrict Application Access<br>Intrusion Detection And Prevention<br>Authenticate Messages<br>Require Multi Factor Authentication<br>Encrypt All Communication Paths<br>Encrypt Device Data<br>Approved Cryptographic Algorithm Used<br>Generate Alerts For Critical Events Or Operations<br>Create Audit Logs<br>Check Message Integrity<br>Validate Inputs<br>Strong Passwords |
| REP HMI | Protect Credentials<br>Restrict Application Access<br>Require Multi Factor Authentication<br>Strong Passwords<br>Non Repudiation Enabled<br>Generate Alerts For Critical Events Or Operations<br>Create Audit Logs |

TABLE IV: Selected devices and their properties, as derived from NESCOR documents.

(and harmful) control commands requires certain smart grid domain knowledge. We represent them on a scale of *High* , *Medium* and *Low*.

- Accessibility: An attacker may be able to physically access some smart grid installation site, and make changes to the system. In some cases, the attacker may have remote access to the facility, e.g., via a stolen account. If the attacker has neither physical nor remote access, we assume she has to launch her attack through public

| Device | List of properties |
|---|---|
| Switch | Enforce Changing Default Configuration<br>Negotiate trunks with other switches<br>Port configured as access port |
| Router | Enforce Changing Default Configuration |

TABLE V: Additional devices and their properties, as defined by CyberSAGE.

| Model parameters | Input | Range |
|---|---|---|
| Skills | IT Skills | High / Medium / Low |
| | Domain Knowledge | High / Medium / Low |
| Accessibility | Logical | True / False |
| | Physical | True / False |
| Intention | Compromise confidentiality | A scale in $[0, 1]$ |
| | Compromise integrity | A scale in $[0, 1]$ |
| | Compromise availability | A scale in $[0, 1]$ |

TABLE VI: CyberSAGE's attacker model.

internet.

- Intention: A threat agent's motivation to launch a attack on the smart grid system is related to the outcome she may wish from the successful compromise. For example, an economic criminal would be more interested in causing a confidentiality compromise as compared to a terrorist whose objective may be to disrupt the normal operations of the smart grid thus making the system unavailable. We capture the level of intention for an attacker to seek different types of security compromise.

Table VI presents the attacker properties included in CyberSAGE's attacker model. We envision that this basic model can be further expanded to provide more comprehensive modeling of various aspects of attackers.

## APPENDIX B
## EXTENSION TEMPLATES FOR DER FAILURE SCENARIOS

The logical relationship between the various input models namely workflow, system and attacker models are discovered via extension templates. We define extension templates as reusable modules that can be used repeatedly to combine heterogeneous pieces of information from our input models in a meaningful and concise manner. The GSA graph generation process is the repeated application of a set of extension templates to produce the final security argument graph. In order to study the NESCOR failure scenario we define a new set of extension templates over and beyond what has been discussed in our previous work [7].

We define 3 graph expansion templates which help us automatically generate the security argument graph. Each of these templates (see templates T1, T2 & T3 below) grow a single vertex $v$ based on the workflow, system and attack models present in $\Sigma$. The resulting star graph $\omega_r$ contains $v$ and at least one additional vertex. Each of the newly added vertices will contain at least one outgoing edge towards $v$. The additional vertices from the star graph $\omega_r$ and the associated edges will be added to the original graph (not shown in pseudocode below). We provide a detailed explanation for template T3 while we present a pseudocode for all the other templates. Extension template T3 is used to decompose a mal-activity step $v$ into all its previous activity steps $\gamma$ obtained by calling the GetLastActivityList($\Sigma$) function. This is because for a activity step $W_f$ to have occurred its previous activity step $W_f - 1$ should also have occurred. We also map $v$ to the component device $v_d$ on which the operation takes place. Depending on the activity step an attacker $v_a$ may arrive to compromise the component device and in process exploit the activity step towards fulfilling her malicious intentions. For example a hacker is more likely to attempt a *network access* exploit as compared to a inept installer who is more likely to use *physical access* as the means to enter into the system. The newly created vertices namely $v_c$, $v_d$, $v_a$ and their associated edges $(v_c, v)$, $(v_d, v)$ , $(v_a, v)$ are added to the returned star graph $\omega_r$.

While formalizing the textual description of the set of 25 DER failure scenarios into mal-activity diagrams we see that mal-activity diagrams share common attack steps. For example, *send command* is a activity step that is shared across all failure scenarios as all have a pro-active step to issue a request to a system component. Similarly in certain scenarios the attacker enters the system either via a *network access* route or via a *physical access* route. The chance that each of these attack steps will succeed depend on the vulnerabilities, mitigations implemented in the device on which the operation is taking place and also on the threat agent's skill to exploit these vulnerabilities. These common attack steps are made into extension templates enabling us to compute the success probability of the attacker to

**procedure** TEMPLATE T1($v$,$\Sigma$)
  $\omega_r \leftarrow$ NewGraph($v$,$\emptyset$,**0**)
  $\gamma \leftarrow$ GetLastActivityList($\Sigma$)
  **for** each $c$ in $\gamma$ **do**
    $dc \leftarrow$ GetMappedActor($c$)
    $v_c \leftarrow$ NewVertex("ActivityStep",$dc$)
    $v_r \leftarrow v_r \cup v_c$
    $E_r \leftarrow E_r \cup (v_c,v)$
    $l(v) \leftarrow l(v_c)$
  **end for**
  **return** ($v$, $\omega_r$)
**end procedure**

Template 1: Template to associate Goal to mal-activity node

**procedure** TEMPLATE T2($v$,$\Sigma$)
  $\omega_r \leftarrow$ NewGraph($v$,$\emptyset$,**0**)
  $at \leftarrow$ GetAttackerDetails($\Sigma$)
  $v_c \leftarrow$ NewVertex("Attacker",$at$)
  $v_r \leftarrow v_r \cup v_c$
  $E_r \leftarrow E_r \cup (v_c,v)$
  $l(v) \leftarrow l(v_c)$
  **return** ($v$, $\omega_r$)
**end procedure**

Template 2: Template to associate mal-activity start node to attacker

exploit these steps. We identify a total of 12 such evaluation extension templates and present 2 of them in the form of pseudo codes below. We believe that by distilling the common attack steps into reusable extension templates the efforts to modelling failure scenarios for other DER components such as AMI are considerably reduced as the new mal-activity diagrams will share a subset of these extension templates.

- *T4 : Physical access template* Certain threat agents such as insiders, installation personal, field technicians may be physical access to the smart grid equipment. The physical accessibility can be abused for tampering the equipment or changing settings,etc and cause the smart grid to enter into unintended failure state. The template T4 represents the chance that such an attack may be possible.
- *T5 : Network access template:* The attack vector via the network access route is a more likely scenario for threat agents as *a)* most smart grid installations are heavily guarded and *b)* it may be possible to leverage a public internet facing interface of the smart grid system such as an office network and then use it to penetrate deeper into the system eventually reaching the secure control system network. This template represents the chance that the attacker can successfully compromise a device by launching network based attacks.

## APPENDIX C
## EVALUATION LOGIC IMPLEMENTED IN OUR EXAMPLE TEMPLATES

**Calculating the success probability of a single attack step.** We first present the quantitative relationship captured in our example extension templates, which describes how to calculate the success probability of an individual attack step based on various relevant factors, including the set of vulnerabilities that can be exploited for launching the attack step, and for each vulnerability, the set of relevant attacker properties and the set of relevant mitigations. To simplify the definition of the quantitative relationship, we make the a number of assumptions in our example extension templates, as detailed below:

- An attack step can be successfully launched by exploiting any one of the relevant vulnerabilities, and these vulnerabilities are independent from each other. Denote the relevant set of vulnerabilities as $vul\_list$. With our assumption, we have:

**procedure** GRAPHEXPANSION($v$,$\Sigma$)
  $\omega_r \leftarrow$ NewGraph($v$,$\emptyset$,**0**)
  $\gamma \leftarrow$ GetPreviousActivityList($\Sigma$)
  **for** each $c$ in $\gamma$ **do**
    $dc \leftarrow$ GetMappedActor($c$)
    $v_c \leftarrow$ NewVertex("ActivityStep",$dc$)
    $v_r \leftarrow v_r \cup v_c$
    $E_r \leftarrow E_r \cup (v_c,v)$
    $l(v) \leftarrow l(v_c)$
  **end for**
  $de \leftarrow$ GetDeviceDetails($v$)
  $v_d \leftarrow$ NewVertex("SystemComponent",$de$)
  $v_r \leftarrow v_r \cup v_d$
  $E_r \leftarrow E_r \cup (v_d,v)$
  $l(v) \leftarrow l(v_d)$
  **if** *v.attacker* $=$ **true then**
    $at \leftarrow$ GetAttackerDetails($\Sigma$)
    $v_a \leftarrow$ NewVertex("Attacker",$at$)
    $v_r \leftarrow v_r \cup v_a$
    $E_r \leftarrow E_r \cup (v_a,v)$
    $l(v) \leftarrow l(v_a)$
  **end if**
  **return** ($v$, $\omega_r$)
**end procedure**

Procedure 3: Graph expansion function to associate a mal-activity step node to its previous activity step, system component and attacker

**procedure** TEMPLATE - PHYSICAL ACCESS(,$\Sigma$)
  $(\mathrm{v}, \omega) \leftarrow Call[Procedure\ GraphExpansion]$
  $v.vul\_list[] \leftarrow$ new vul_list["Physical access may be obtained by unauthorized individuals"]
  $v.vul\_list[0].addAttackerProperties$("Physical accessibility","True")
  $v.vul\_list[0].addRelevantMitigation$("Restrict physical access"]
  $v.device \leftarrow$ GetDevice($\Sigma$)
  $v.attacker \leftarrow$ GetAttacker($\Sigma$)
  $v.eval \leftarrow$ attackStepSuccessProb(vul_list[],device,attacker)
  **return** $(v, \omega)$
**end procedure**

Template 4: Physical Access Template

$$p_{\text{attack step}} = 1 - \prod_{vul \in vul\_list} (1 - p_{vul})$$

where $p_{\text{attack step}}$ is the probability for the attack step to succeed, and $p_{vul}$ is the probability that the attacker exploits a specific vulnerability $vul \in vul\_list$ and successfully execute the attack step.

- The probability $p_{vul}$ for each individual vulnerability $vul$ depends on the relevant attacker properties (which we denote as $A_{vul}$) and the relevant mitigations (which we denote as $M_{vul}$).
- We first consider the case when there is no mitigation in place at all. Let us denote the success probability to exploit a particular vulnerability $vul$ under this assumption as $p'_{vul}$, which only depends on an attacker's properties, such as its physical access privilege, IT skill, and domain knowledge. The exploitation of different

**procedure** TEMPLATE - NETWORK ACCESS($\Sigma$)
  $(\mathrm{v}, \omega) \leftarrow Call[Procedure\ GraphExpansion]$
  $v.vul\_list[] \leftarrow$ new vul_list["Commands or messages may be inserted into the network"]
  $v.vul\_list[0].addAttackerProperties($"Network accessibility", "True"$)$
  $v.vul\_list[0].addAttackerProperties($"Domain skills", "[(High, 0.9), (Medium, 0.7), (Low, 0.4)]"$)$
  $v.vul\_list[0].addAttackerProperties($"IT skills", "[(High, 0.99), (Medium, 0.8), (Low, 0.5)]"$)$
  $v.vul\_list[0].addRelevantMitigation($"Restrict network access", "Enforce restrictive firewall rules"$]$
  $v.device \leftarrow$ GetDevice($\Sigma$)
  $v.attacker \leftarrow$ GetAttacker($\Sigma$)
  $v.eval \leftarrow$ attackStepSuccessProb(vul_list[],device,attacker)
  **return** $(v, \omega)$
**end procedure**

Template 5: Network Access Template

vulnerability can depend on different attack properties. For example, the exploitation of the vulnerability "Physical access may be obtained by unauthorized individuals to DER settings through the system UI", depends on the attacker property of "physical access privilege", but not on other properties like "IT skill". In comparison, the exploitation of the vulnerability "Commands or other messages may be inserted on the network between the user interface and the DER system, that result in unauthenticated changes to sensitive parameters", can depend on two attacker properties, "IT skill" and "domain knowledge", but may not depend on the "physical access". To simplify the way the user can quantitatively specify such dependency, we restrict each attacker property to take value from a small number of discrete levels, e.g., for "physical access", it has just two levels, "Yes" or "No"; for "IT skill" and "domain knowledge", they both have three levels, "High", "Medium", and "Low". We will then require the users to define $p'_{vul}$, the success probability to exploit vulnerability $i$ when there is no mitigations, based on all combinations of these relevant attacker properties. Table VII and Table VIII give two example set of inputs a user need to provide. Specifically, Table VII shows the user input for a vulnerability that depends on a single property, and Table VIII shows the user input for a vulnerability that depends on two properties.

| physical access privilege | |
|---|---|
| yes | 0.9 |
| no | 0.01 |

TABLE VII: $p'_{\text{physical access vulnerability}}$, which depends on a single attacker property

| | Domain knowledge | | |
|---|---|---|---|
| IT skill | high | medium | low |
| high | 0.81 | 0.72 | 0.63 |
| medium | 0.45 | 0.4 | 0.35 |
| low | 0.09 | 0.08 | 0.07 |

TABLE VIII: $p'_{\text{network access vulnerability}}$, which depends on two attacker properties.

| | IT skill | Domain knowledge |
|---|---|---|
| high | 0.9 | 0.9 |
| medium | 0.5 | 0.8 |
| low | 0.1 | 0.7 |

TABLE IX: $q_{textrmITskill}$ and $q_{textrmDomainknowledge}$ for generating $p'_{\text{network access vulnerability}}$.

To reduce the number of user inputs required in the second case where there are multiple relevant attacker properties, we further assume these attacker properties affect the success probability in an independent manner, and we define $q_j$ for each relevant attacker property $j \in A_{vul}$, such that $p'_{vul} = \prod_j q_j$. With this assumption, we can replace the definition in the Table VIII by Table IX. Comparing Table VIII with Table IX, the number of

required user inputs is reduced from $3 \times 3 = 9$ to $3 + 3 = 6$. This savings can be significant when the number of relevant attacker properties is large.

- Now we consider the effectiveness of mitigations. Recall that each vulnerability $vul$ has a set of relevant mitigations $M_{vul}$. We assume each mitigation takes effect in an independent way. We assume that user can provide good estimation of the effectiveness of each mitigation, as measured by the the scale that the mitigation can reduce the probability of successful vulnerability exploitation. Hence, for each mitigation $k \in M_{vul}$, we will ask the user to define its probability-reduction scale, $r_k \in [0, 1]$ (the lower the value of $r_k$, the more effective the mitigation $k$ in reducing the security risk of vulnerability $i$). This will be defined as a property in the corresponding device. With a given set of mitigation inputs, the probability for the vulnerability to be exploited becomes $p_{vul} = p'_{vul} \times \prod_{k \in M_{vul}} r_k$.

To summarize the above discussions (with all assumptions made), the final equation the extension template will use to calculate $p_{attackstep}$ is:

$$p_{\text{attack step}} = 1 - \prod_{vul \in vul\_list} (1 - \prod_{j \in A_i} q_j \times \prod_{k \in M_i} r_k)$$

To calculate the success probability of a single attack step, we will need to solicit the following inputs from the user:

- The set of relevant vulnerabilities $vul\_list$, which may consist of one or multiple vulnerabilities.
- For each vulnerability $vul$, the set of relevant attacker properties $A_{vul}$.
- For each attacker property $j \in A_{vul}$, its range of possible values, and for each value, the corresponding $q_j$.
- For each vulnerability $vul$, the set of relevant mitigations $M_{vul}$.
- For each mitigation $k \in M_{vul}$, its probability-reduction value $r_k$.

The pseudo code in Procedure 6 summarizes the main evaluation logic we implemented for example extension templates.

```
1:  procedure AttackStepSuccessProb(vul_list[], device, attacker)
2:      p = 1;
3:      for each vul in vul_list[] do
4:          q = 1; r = 1;
5:          for each j in vul.relevant_attacker_property_list[] do
6:              q = q × j.probability[attacker.property_list.get_value(j.name)];
7:          end for
8:          for each k in vul.relevant_mitigation_list[] do
9:              r = r × device.property_list.get_value(k.name)];
10:         end for
11:         p = p × (1 − q × r);
12:     end for
13:     return 1 − p;
14: end procedure
```

Procedure 6: AttackStepSuccessProb function

**Calculating the success probability of the whole mal-activity diagram.** The success probability of individual attack steps are then combined over our security argument graph according to the logical OR / AND relationship among them. We use LibDAI to deal with shared variables. See more details about how this combination is computed in our earlier work [7].

We further introduce here the parameters to characterize the malicious intent of the attacker, and gives three probabilities for an attacker to launch attacks that related to C / I / A respectively. With this, for a given mal-activity diagram, $p_{\text{failure scenario}}$ calculated from the mal-activity diagram should be further multiplied by $p_{\text{intention to launch specific mal-activity}}$. This is intended to capture the case where an attacker may be able to launch the attack, but she doesn't have the intention to do so.

## APPENDIX D
## SENSITIVITY ANALYSIS FOR DER.1 FAILURE SCENARIO

Level of sensitivity to inputs is a crucial factor for the accuracy of a rule based system. Therefore, we have conducted some experiments on CyberSage to analyse the input sensitivity of its rule engine. Table:X contains the experiment results based on the first Failure Scenario, DER.1, of the NESCOR Recommendation Document.

| | Baseline | Physical Accessability | Network Accessability | Restrict Both |
|---|---|---|---|---|
| $P = P_{Init}$ **i.e. (Initial Inputs)** | | | | |
| Inep Install | 0.12 | 0.04 | 0.12 | 0.04 |
| Hacker | 0.71 | 0.71 | 0.61 | 0.61 |
| Indus. Spy | 0.87 | 0.87 | 0.84 | 0.84 |
| $P = P_{min}$ **i.e. (90% of** $P_{init}$**)** | | | | |
| Inep Install | 0.08 | 0.02 | 0.08 | 0.02 |
| Hacker | 0.53 | 0.53 | 0.41 | 0.41 |
| Indus. Spy | 0.72 | 0.72 | 0.63 | 0.63 |
| $P = P_{max}$ **i.e. (110% of** $P_{init}$**)** | | | | |
| Inep Install | 0.17 | 0.07 | 0.17 | 0.07 |
| Hacker | 0.83 | 0.83 | 0.77 | 0.77 |
| Indus. Spy | 0.92 | 0.92 | 0.91 | 0.91 |

TABLE X: Probability Results for different input values

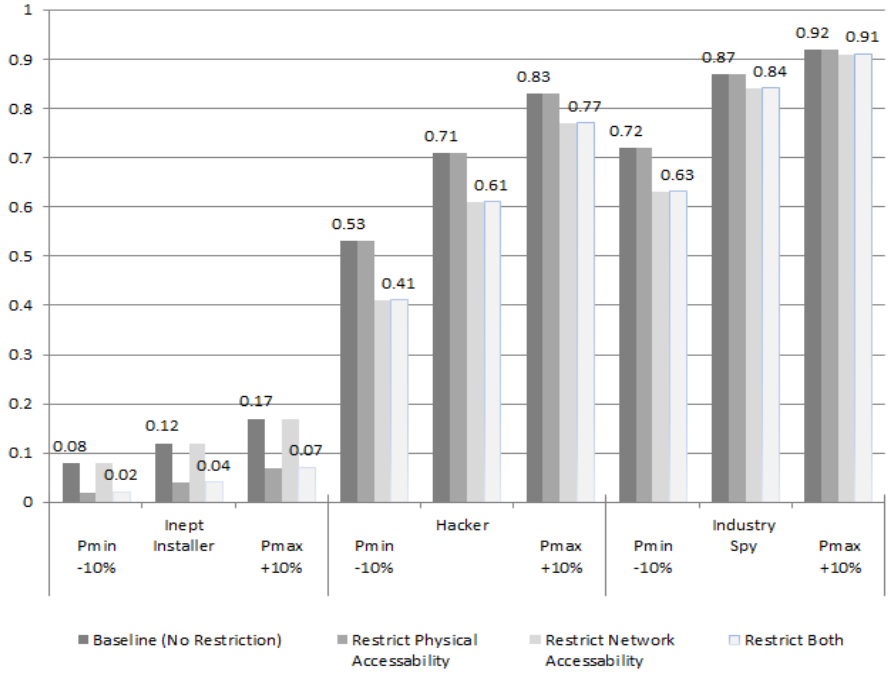


Fig. 5: Probability Result Chart for different input values (P)

As shown in the above table, firstly we define three types of attack profiles with different intensions and skills, as Inept Installer, Hacker and a Industry Spy [Table:II,III]. There are two mitigations considered in the above example. i.e. Restrict Physical Accessibility, Restrict Network Accessibility. Initially, the experiment was started with no

mitigations. Then, we have added the Physical Accessibility Restriction and Network Accessibility Restriction, one after the other. After obtaining the probability values for all the combinations of attack profiles and mitigations, the experiment was repeated after reducing the input values by -10% and increasing by +10%. (Refer to Table:**??** for input values). If we consider $P$ as the initial probability inputs, we can calculate the $P_{min}$ as $P_{min} = 90\%.P$ and the $P_{max}$ as $P_{max} = Min(110\%.P, 1)$. The impact of the input value changes can be considered as a level of sensitivity of the rule engine.

The bar chart in the Figure:5, shows how the probability output increases with the input value changes from $P_{min}$ to $P_{max}$ in each attacker profiles. Lowest and highest probability values were shown as labels in each bar-group to indicate the effect mitigations in the output. Eventhough there is a uniform increase in the output for Inept Installer and Hacker, Industry Spy shows low increase when the probaability is very near to 1. This is the convergent effect due to the upper limit of the probability. Moreover, when the attacker has a profile of very strong skills, then adding mitigations has less effect on the output probability.

| | **Baseline (%)** | **Restric Physical Accessability (%)** | **Restrict Network Accessability (%)** | **Restrict Both (%)** | **Average (%)** |
|---|---|---|---|---|---|
| $P = P_{min}$ **i.e. (90% of** $P_{init}$**)** | | | | | |
| Inep Install | 33.33% | 50.00% | 33.33% | 50.00% | 41.67% |
| Hacker | 25.35% | 25.35% | 32.79% | 32.79% | 29.07% |
| Indus. Spy | 17.24% | 17.24% | 25.00% | 25.00% | 21.12% |
| $P = P_{max}$ **i.e. (110% of** $P_{init}$**)** | | | | | |
| Inep Install | 41.67% | 75.00% | 41.67% | 75.00% | 58.33% |
| Hacker | 16.90% | 16.90% | 26.23% | 26.23% | 21.57% |
| Indus. Spy | 5.75% | 5.75% | 8.33% | 8.33% | 7.04% |

TABLE XI: Sensitivity Level Percentage

In order to analyse the sensitivity level based on a percentage difference of the output, we have derived the Table:XI. According to the Chart in Figure:6, significant increase in the overall probability differences can be observed for Inept Installer, whereas the Industry Spy shows low differences. Hence, the input value change has high impact on the Inept Installer (Low Skilled Attacker Profiles) than the Industry Spy ( Highly Skilled Attack Profiles). However, when we reduce the probability, sensitivity level increases for Hackers and Industry Spies, which is somewhat opposite to the behaviour of Inept Installer. Furthermore, outof all the sensitivity levels, the lowest figure can observed in the $P_{max}$ of the Industry Spy. The main reason for this observation is due to the upper limit ($0 <= P <= 1$) of any probability inputs. (I.e. Probability difference will be reduced when the output is getting close to 1.) Based on these experimental results, we can conclude that the overall sensitivity has a negative relation with the level of skills of the attacker. Having the correct level of understanding on the sensitivity of input value changes, a security expert can define the input probability values appropriately. Multiple, iterations of fine-tuning is essential for for highly accurate results. Even though it would be a time consuming task, final result can be reused to obtain promising results on multiple Smart Grid Systems. Based on this approach, CyberSage can have a well optimized rule engine with knowledge base of Cyber Security, as a set of probability values defined for multiple set of attacker-mitigation combinations.

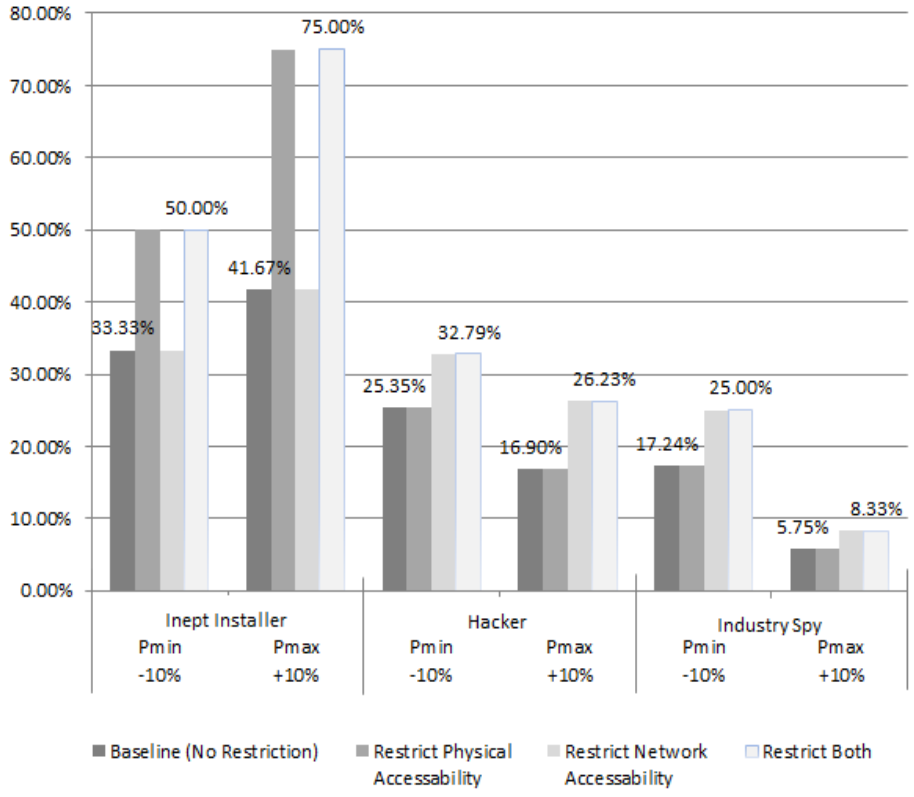


Fig. 6: Sensitivity Analysis Based on Difference%