Modeling Mission Impact of Cyber Attacks on Energy Delivery Systems

Md Ariful Haque[1(✉)], Sachin Shetty[1], Charles A. Kamhoua[2], and Kimberly Gold[3]

[1] Department of Computational Modeling and Simulation Engineering, Old Dominion University, Norfolk, VA, USA
mhaqu001,sshetty@odu.edu
[2] Network Security Research, The U.S. Army Research Laboratory, Adelphi, MD, USA
charles.a.kamhoua.civ@mail.mil
[3] Naval Surface Warfare Center, Crane Division, Crane, IN, USA
kimberly.gold@navy.mil

Abstract. Today energy delivery systems (EDS) face challenges in dealing with cyberattacks that originate by exploiting the communication network assets. Traditional power systems are highly complex and heterogeneous. These systems focus on reliability, availability, and continuous performance and, thus, not designed to handle security issues. Network administrators often utilize attack graphs to analyze security in EDS. Although attack graphs are useful tools to generate attack paths and estimate possible consequences in a networked system, they lack incorporating the operational or functional dependencies. Localizing the dependencies among operational missions, tasks, and the hosting devices in a large-scale cyber-physical network is also challenging. Current research works handle the system dependency and the attack scenario modeling separately using dependency graphs and attack graphs, respectively. To address the gap of incorporating the mission operational dependencies with possible attack scenarios, in this work, we offer an approach to assess the cyberattack impact on the operational mission of the EDS by combining the logical attack graph and mission functional dependency graph. We provide the graphical modeling details and illustrate the approach using a case study of SCADA (supervisory control and data acquisition) operations within an EDS environment.

Keywords: Energy delivery systems · Attack graph · Mission dependency · Impact propagation graph · Impact assessment · Operability

1 Introduction

The energy delivery systems (EDS) increasingly rely on the communication network for monitoring, operation, and control. The EDS broadly divides itself

N. Park et al. (Eds.): SecureComm 2020, LNICST 336, pp. 41–61, 2020.
https://doi.org/10.1007/978-3-030-63095-9_3

into the physical, control, and cyber layer. The physical layer is responsible for supporting power generation, transmission, and distribution. The control layer is responsible for sensing and reporting field-level data and corrects the set-points as necessary through automated or manually initiated commands. At the same time, the cyber layer comprises the communication networks to monitor and evaluate performances of the physical layer devices and business-level operations. EDS's key component is the supervisory control and data acquisition (SCADA) system, which provides facilities to monitor, report, and controls different types of physical processes simultaneously. Therefore, the power system's reliable operation is heavily dependent on the attack-resilient functioning of the SCADA and the associated cyber systems.

Because of the heterogeneity and complex interconnectivity among the cyber and physical layers, the energy systems face challenges in assessing the overall mission impact of cyberattacks originating from the cyber layer. There are two research questions as yet not entirely analyzed by the research community:

1. How could we incorporate the critical operational dependencies to the traditional security analysis models to get a comprehensive assessment of mission impact and cyber resilience due to a potential exploit? Here, by dependencies, we mean mapping of the business mission to operation-critical functions/tasks, tasks to software programs/applications/services, and services to the underlying devices.
2. To develop preventive resilience methodologies, how could we assess and quantify the effects of cyberattacks on the operations of the physical layer, which comes from the stepping-stone exploitation of the cyber layer?

To identify the potential exploitable attack paths in the enterprise networks and CIs, researchers often rely on the analysis based on attack graph techniques. Although attack graphs provide insights in analyzing the network security flaws, they do not incorporate the mission-specific operational dependencies while depicting the attack paths. Thus, there remains a gap in evaluating overall mission impact and system operability during adverse attack scenarios. To address the gap of incorporation of mission dependency in attack graph analysis, Sun et al. [1] propose a technique to model the dependency relations by utilizing the service dependency graph and attack graph. Cao et al. [2] present a quantitative metric for business process impact assessment for Enterprize networks using attack graphs and entity dependency graphs. Both the works give a concrete reason to incorporate the dependency into the attack graph. Still, the works lack in providing formal guidance on how to correlate the attack graph and dependency graph; thus, it is not clear how to apply the methods in a specific cyber-physical systems environment.

In this work, we have addressed some of the above research gaps. We introduce new graph types specific to mission, more specifically, mission functional dependency graph, mission impact propagation graph, and mission impact assessment graph. We have developed a NIST defense-in-depth [3] architecture-based SCADA operational case study illustrating the approaches that we formalize under the context of this paper. The main contributions of this work are as follows:

i) A graph-theoretic modeling approach to incorporate the critical mission dependencies (e.g., mapping the mission to tasks, tasks to applications, and applications to hosts) to the logical attack graph model
ii) Quantifying the overall mission impact of cyberattacks and operability of EDS originating by exploiting the communication network assets and
iii) A realistic SCADA operations case study to systematically illustrate the modeling approach and ways to implement.

We organize the rest of the paper as follows. Section 2 presents the definitions and assumptions. Section 3 describes the system model for mission impact assessment. Section 4 explains the modeling approach in detail. Section 5 illustrates the case study and analysis. Section 6 provides some insights on the related works. Finally, Section 7 concludes the article with future directions.

2 Preliminaries: Definitions and Assumptions

Definition 1. ***Mission:*** *A mission is an objective to maintain the operational functionality of the constituent parts of the system through securing the performance of the required tasks according to the design or specifications. A mission relies on a set of functions termed as 'tasks' to get accomplished.*

Definition 2. ***Task:*** *A task is the component of a mission that has its specific function/functions to be carried out to maintain the proper operational level of the underlying programs/processes/devices.*

Definition 3. ***Application:*** *An application is an individual or combined licensed or open-source software (commonly known as 'programs') in use by the network that supports the functions of the task. Here, we use the term 'application' instead of 'program' or 'service' to mean the same.*

Definition 4. ***Host:*** *A host is a network device that houses programs or software applications. The types of hosts include but not limited to:*

- ***IT Devices:*** *Servers, desktops, databases, other computing devices, etc.*
- ***Network Devices:*** *Firewalls, routers, switches, Wi-Fi access points, etc.*
- ***SCADA Devices:*** *Historians, human-machine interfaces (HMI), engineering workstations, master terminal units (MTU), etc.*
- ***Physical Controllers:*** *PLC (programmable logic controller), RTU (remote terminal unit), IED (intelligent electronic device), PMU (phasor measurement unit), sensors, transducers, actuators, etc.*

Definition 5. ***Operability:*** *Operability is the state of a host functioning at some level of performance. The unit of operability is equivalent to a von Neumann-Morgenstern utility measure (expressed in utils as in FDNA [4]). We denote here the utils as a real-valued number within the interval [0,1].*

Definition 6. ***Present Operational Capability (POC):*** *POC is a time-dependent operational capability of the host indicating the operability level of the host/device at the given time instant. POC 1.0 means fully operable and POC 0.0 means fully inoperable.*

Definition 7. ***Strength of Dependency (SOD):*** *It depicts to what extent the operability level of the parent node can influence the child node's operability level. For example, an SOD between a task and a host indicates the extent/fraction of the operability level of the host that can be reduced due to the malfunction/compromise/impact on the task.*

Definition 8. ***Impact Factor:*** *'Impact' means the effect or consequence of an event, incident, or occurrences on the operability of the constituent parts of the system or network. The impact factor is the fraction or amount of loss or reduction in the associated device's performance/operability.*

Assumption 1. Paths between single component nodes of mission graphs are acyclic. For example, if there is an inter-dependency of task t_1 on application a_1, then there is no reverse path of dependency from a_1 to t_1. Similarly, if there is an intra-dependency of task t_1 to t_2, then there exists only one path from $t_1 \rightarrow t_2$, and no paths from $t_2 \rightarrow t_1$.

3 System Mission Impact Assessment Model

We provide a high-level mission impact assessment model in Fig. 1. The input system comprises network vulnerability and mission dependency data. We utilize open-source network scanning tools (such as Nessus[1], OpenVAS[2], etc.) to collect

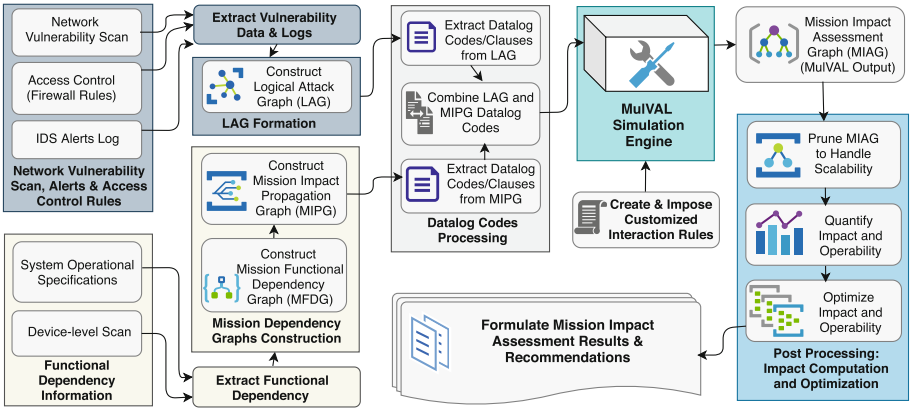


Fig. 1. System mission impact assessment model

[1] NESSUS Vulnerability Assessment (https://www.tenable.com/products/nessus).
[2] OpenVAS - Open Vulnerability Assessment Scanner (https://www.openvas.org/).

the network vulnerability information. We use Snort for IDS (intrusion detection system) alerts. There are other vulnerability scanning tools available specific to the ICS environment, such as Nextnine's ICS shield, Radiflow, Darktrace ICS, and Splunk, etc. We extract the required network logs from vulnerability scan, IDS alert, and firewall rules to generate the logical attack graph (LAG). We use quantitative score of the vulnerabilities from CVSS[3] and NVD[4]. We utilize host-level scanning and system operational documents to identify the mission-critical operational tasks and their dependencies on the devices. There are some open-source/commercial tools to identify the network application-level dependencies, such as ManageEnige's application discovery and dependency mapping (ADDM), Device42, AppDynamics, etc. The extracted system dependency data leads to the modeling of the mission functional dependency graph (MFDG). We build the mission impact propagation graph (MIPG) by reversing the edges and using transitivity rules. We combine the logical attack graph (LAG) and mission impact propagation graph (MIPG) using Datalog[5] clauses. We also define and add our custom interaction rules for the open-source MulVAL [5] security analyzer. The MulVAL provides the mission impact assessment graph (MIAG) as an output. Based on the network size, we may need to prune the mission impact assessment graph to handle the scalability issue. We then compute the tasks' impacts and relate that to the operability of hosts and impact on the mission as we provide in Subsect. 4.6 and 4.7.

Let us consider an example of the operation & maintenance of a manufacturing plant (i.e., the mission). The process depends on real-time data collections by the sensor devices, and a historian device stores those data (i.e., the task). The historian is of 'SIEMENS,' and the historian software application is 'SIMATIC 2014 SP3 Basic' (i.e., the app). This application is installed and running in a Windows Server 2008 SP2 (i.e., the host). The Windows Server software has a 'remote code execution' vulnerability with the ID 'CVE-2017-0148'. In this case, the attack graph depicts a vulnerability that would lead to the privilege of code execution in the Windows Server. The **'Network Vulnerability'** block captures these pieces of information in the above diagram. Now, the operational mission depends on data collection and storing tasks, which again depend on the 'SIMATIC' application, which is hosted by Windows-based Server. The MFDG captures these dependency scenarios. Therefore, we find that if there is an executive privilege on the host (i.e., Windows Server), then this privilege would lead to the impact on the associated applications, tasks, and propagate to the overall mission. MIPG captures this scenario, which we construct by reversing the MFDG and using transitivity rule. The block **'Mission Dependency Graph Construction'** reflects this whole dependency scenario. Finally, we consolidate the Datalog codes and use custom made interaction rules in the MulVAL simulation. We present detailed formal definitions of the graphs in Sect. 4.

[3] Common Vulnerability Scoring System (https://www.first.org/cvss/).
[4] National Vulnerability Database (https://nvd.nist.gov/).
[5] Declarative Logic Programming.

4 Graphical Modeling Approach Details

In this section, we provide formal definitions of the graph-theoretic modeling approaches and computation processes to derive the mission impact and system operability.

4.1 Logical Attack Graph (LAG)

We utilize the logical attack graph definition by Gonda et al. [6]. The LAG has three types of nodes: (1) ***Primitive fact nodes*** (N_c) represent **facts** about the system, such as network connectivity, access control or firewall rules, user accounts on host machines, etc.; (2) ***Derivation nodes*** (N_e) also known as **rule/action/execution/exploit** nodes represent an action the attacker can take or satisfy certain conditions to gain a privilege in the system; (3) ***Derived fact nodes*** (N_p) (also known as ***privilege nodes***) represent a **capability** an attacker gains after performing an action by satisfying the pre-conditions. There are three relationships possible among the nodes. An **'AND'** relation implies that all the pre-conditions need to be satisfied; an **'OR'** relation means that either of the pre-conditions to be fulfilled; a **'FLOW'** relation suggests that the information flows its effects to the successor node. In a LAG, we have:

- Set of vertices, $V = N_c \cup N_e \cup N_p$
- Set of edges, $E \subseteq (N_e \times N_p) \cup ((N_p \cup N_c) \times N_e)$
- If the set of all existing vulnerabilities is V', then vertex intrinsic weight,

$$f_v = \begin{cases} 1.0 & \text{if} v \in N_c \cup N_p \text{ and } v \notin V' \\ 0.8 & \text{if} v \in N_e, \text{ i.e., } 80\% \text{ probability of exploit as in } MulVAL \\ \frac{CVSS_v}{10} & \text{if} v \in N_c \text{ and } v \in V' \end{cases}$$

4.2 Mission Functional Dependency Graph (MFDG)

Definition 9. ***A mission functional dependency graph*** G_b *is a directed acyclic graph denoted by* $G_b = (N_m, N_t, N_a, N_h, E, f_i, w_e, L, \alpha, \gamma)$ *where* N_m, N_t, N_a, *and* N_h *represent mission, task, application, and host nodes, respectively;* E *is a set of edges denoted as* (u, v) *that represents direction of dependency;* f_i *is a non-negative weights associated with the nodes representing the intrinsic operability of the nodes;* w_e *is the edge weights;* L *is a mapping of vertices to the type (AND, OR, FLOW) of logical dependence among the vertices (i.e., mission, task, application, or host);* $\alpha : N \times N \rightarrow [0, 1]$ *is the score assignment function representing strength of inter-dependency (inter-SOD). Similarly,* $\gamma : N \times N \rightarrow [0, 1]$ *represents strength of intra-dependency (intra-SOD).*

- Set of vertices, $V = N_m \cup N_t \cup N_a \cup N_h$
- If the task t depends on application a (i.e., $t \rightarrow a$), and host h is housing application a (means, $a \rightarrow h$), then using the *transitivity rule*, we can say, $t \rightarrow h$, which means task t has dependency on host h. Thus, we can prune the MFDG to remove the application nodes and establish direct dependency between task to host.

– $N_m \times N_a = \emptyset$, and $N_m \times N_h = \emptyset$; this indicates there is no direct dependence of the mission on the applications or hosts.

4.3 Mission Impact Propagation Graph (MIPG)

We generate the mission impact propagation graph (MIPG) by applying a recursion on the pruned MFDG. Formally, we define the MIPG as follows.

Definition 10. ***A mission impact propagation graph.*** *G_c is a directed acyclic graph represented by the tuple $G_c = (N_{hp}, N_{it}, N_{im}, N_d, E, f_v, w_e, \alpha, \gamma, h)$; node N_{hp} represents an execution privilege on host; N_{it} is task impact node; N_{im} represent mission impact node; N_d represents rule node to satisfy to propagate the effect to child node; E is a set of edges denoting the direction of dependency; f_v is the node intrinsic score as in LAG; w_e is the edge weights computed using the logical dependence; α is the inter-SOD; γ is the intra-SOD, and h is a mapping of vertices to the type (i.e., AND, OR, FLOW) of dependence.*

– Set of vertices, $V = N_{hp} \cup N_d \cup N_{it} \cup N_{im}$
– Set of edges, $E \subseteq (N_{hp} \times N_d) \cup (N_d \times N_{it}) \cup (N_{it} \times N_d) \cup (N_d \times N_{im})$

4.4 Integrating LAG and MIPG Using Subgraph Merging Technique to Generate MIAG

We apply here a subgraph merging technique to combine LAG and MIPG. First, we present the types of edges in LAG and MIPG to help the audience understand the merging concepts.

Types of Edges in LAG. There are three types of edges in the LAG as shown in Fig. 2a. An edge (c, e) connecting primitive node (i.e., preconditions) (N_c) to the exploit (i.e., action) node (N_e) implies that by satisfying the condition c an attacker can execute exploit e. An edge (e, p) that connects an exploit node (N_e) to a derived node (i.e., privilege node) (N_p) means that by exploiting e an attacker can gain privilege p. An edge (p, e) from a derived node (N_p) to a exploit (i.e., action or rule) node (N_e) states that p is again a precondition to next exploit e.

Types of Edges in MIPG. In Fig. 2b(1), a gained host execution privilege p leads to exploit the next dependency relation d. In Fig. 2c(2), the dependency relation d implies flow of impact i on the next task or mission node (N_{it}, *or* N_{im}).

Subgraph Merging. We apply here a subgraph merging techniques utilizing the graph union operation [7]. We consider the LAG as the first subgraph and MIPG as the second subgraph. We then apply the graph union operations on LAG and MIPG to build the combined MIAG. We only consider the nodes and edges for this illustration as the other attributes are same for both LAG

and MIPG. Formally, if *LAG* $G_a := G(V_1, E_1)$ where $V_1 = N_c \cup N_e \cup N_p$ and $E_1 \subseteq (N_e \times N_p) \cup ((N_p \cup N_c) \times N_e)$ and MIPG $G_c := G(V_2, E_2)$ where $V_2 = N_{hp} \cup N_d \cup N_{it} \cup N_{im}$ and $E_2 \subseteq (N_{hp} \times N_d) \cup (N_d \times N_{it}) \cup (N_{it} \times N_d) \cup (N_d \times N_{im})$, then we can utilize the sub-graph union operator ($\oplus$) as follows with the help of the mapping function $\lambda : N_p \mapsto N_{hp}$. The mapping function λ indicates that the privilege node N_p (i.e. execution privilege p) in LAG maps to host exploitation node N_{hp} (which is also execution privilege p) in MIPG. Thus, we can formulate the mission impact assessment graph MIAG as $G_d = G(V, E) := G_a \oplus G_c$, where G_a and G_c are the LAG and MIPG respectively. Then, $V := V_1 \oplus V_2 = V_1 \cup (V_2 \setminus u) = N_c \cup N_e \cup N_{hp} \cup N_d \cup N_{it} \cup N_{im}$, $u = N_{hp}$. Also, $E := E_1 \oplus E_2 = E_1 \cup E_2$.

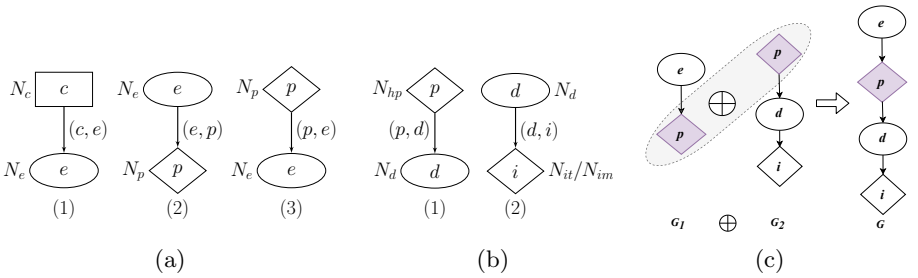


Fig. 2. (a) Edge types in LAG, (b) Edge types in MIPG, (c) Applying subgraph merging technique on LAG and MIPG to get MIAG (Subsect. 4.5)

4.5 Mission Impact Assessment Graph (MIAG)

Definition 11. ***A Mission Impact Assessment Graph (MIAG)*** *is a directed acyclic graph represented as* $G_d = (V, E, f_v, h, \alpha, \gamma, \lambda)$ *where* V *is a set of vertices that represents host information, pre-conditions, vulnerabilities, exploits, impacts on tasks, and impact on missions;* E *is a set of edges;* f_v *is a non-negative weights associated with the vertices;* h *is a mapping of vertices to the type (i.e., AND, OR, FLOW) of logical dependence among the vertices;* α *represents inter-SOD;* γ *represents intra-SOD;* $\lambda : p \mapsto hp$ *is a mapping function that maps an privilege* p *in LAG to the host privilege* hp *in MIPG.*

4.6 Impact Score Quantification Process Using MIAG

We associate each node (V) in the MIAG with two scores; the intrinsic score f_v and the derived score $P : V \rightarrow [0, 1]$. The intrinsic score stands for the inherent likelihood of an action/exploit e to execute, given that all the pre-conditions required for performing e in the given attack sequence are satisfied. The derived score measures the overall likelihood that an attacker successfully reaches and execute the exploit e and gain privilege p. We assume here that the events that an attacker may execute different exploits are independent of each other for simplicity. For shortening the illustration, we only consider conditions, exploits,

and privilege nodes in MIAG to illustrate the computation process, which we also extend for the task and mission nodes. Given an MIAG, we formalize the derived scores for exploit and privilege $P(e)$ and $P(p)$ respectively as below:

- $P(e) = f_v(e) \cdot \prod_{c \in R_e} P(c)$, where $R_e \subseteq (N_c \times N_e) \cup (N_p \times N_e)$
- $P(p) = f_v(p)$, $if\ R_p(p) = \emptyset$, and $P(p) = f_v(p) \cdot \bigotimes_{e \in R_p(p)} P(e)$ otherwise; where $R_p(p) \subseteq (N_c \times N_p) \cup (N_e \times N_p)$, and the operator $\bigotimes P(e) = P(e_1) \cup P(e_2) = P(e_1) + P(e_2) - P(e_1) \cdot P(e_2)$, where $\{e_1, e_2\} \subseteq R_p(p)$.

Similar computation process we apply for computing the derived scores/ impacts of task nodes.

4.7 Computing Device and System Operability Using Dependency Relations and Task Impacts

We present here six different cases to compute the operability considering the inter and intra-dependency relations among tasks and hosts. Here, we utilize FDNA by Garvey et al. [4] and mission impact assessment by Jakobson et al. [8] to formulate the impact factor and the operability. Here, we use the notations as follows: h_n denotes nth host/device; t_n indicates nth task; I_{t_n} is the impact on task t_n found from MIAG; OC_{h_n} is the computed operability of host h_n; $POC_{h_n}(t)$ depicts present operational capability before any attack incident; $IF_{h_n}(t^+)$ is the impact factor (i.e., the fraction of reduction in operability) on host h_n at time t^+; t^+ are discrete time instances, where $t^+ > t$. There are two different dependency relations: inter-dependency and intra-dependency. $\alpha_{ij} = f(OC_{t_i}, POC_{h_j})$ is the strength of inter-dependency (inter-SOD) (having value in $[0, 1]$) of task t_i on host h_j which is a function of operability of task t_i (i.e, OC_{t_i}), and present operational capability of host h_j (i.e., POC_{h_j}); Similarly, $\gamma_{ij} = g(OC_{t_i}, OC_{t_j})$ is the intra-SOD (having value in $[0, 1]$) of task t_i on another task t_j which is a function of the operability of task t_i (i.e, OC_{t_i}), and task t_j (i.e, OC_{t_j}). We can formulate these values utilizing the feeder-receiver dependency methods described by Guariniello et al. [9]. In case, if we have no way to measure α_{ij}, and γ_{ij}, we can utilize the values of 1.0, 0.5, and 0 for *fully-dependent*, *partially-dependent*, and *non-dependent* cases respectively.

Case I: 'FLOW' Inter-dependency. In Fig. 3(a1) we have a task t_n 'FLOW' depends on the host machine h_n with a strength of dependency between them α_n. Figure 3(a2) shows their dependency in the form of MIDG. Here the impact on task t_n (i.e., I_{t_n}) has a 'FLOW' dependency on the operability of host h_n (i.e., OC_{h_n}). We can compute the operability OC_{h_n} at a discrete time instant t^+ using the present operational capability of h_n (i.e., POC_{h_n}) before the attack incident (i.e., time instant t) and the flow of task impact as follows, where $t^+ > t$.

$$IF_{h_n}(t^+) := \alpha_{nn} I_{t_n}(t^+) \tag{1}$$

$$OC_{h_n}(t^+) := max(POC_{h_n}(t) - IF_{h_n}(t^+), 0) \tag{2}$$

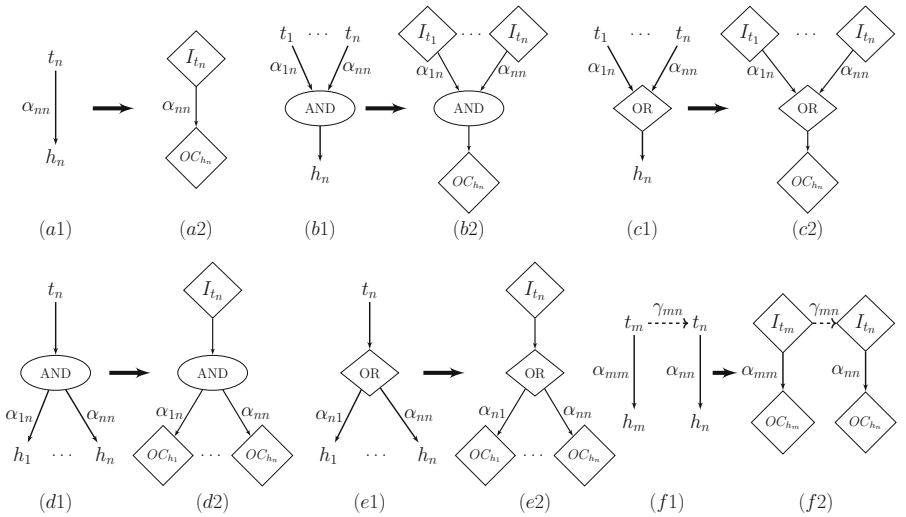


Fig. 3. Dependency relations and operability computation reference figure; (a1) and (a2) task to host 'FLOW' dependency; (b1) and (b2) multiple tasks 'AND' depend on single host; (c1) and (c2) multiple tasks 'OR' depend on single host; (d1) and (d2) single task 'AND' depend on multiple hosts; (e1) and (e2) single task 'OR' depend on multiple hosts; (f1) and (f2) task to task intra-dependency

$$POC_{h_n}(t^+) = OC_{h_n}(t^+) \tag{3}$$

In Eq. 1, we compute the impact factor at time t^+. We utilize $IF_{h_n}(t^+)$ in Eq. 2 to compute $POC_{h_n}(t^+)$ which is updated operational capability of host h_n after the attack incident reduces the current operability; In Eq. 3, the $POC_{h_n}(t^+)$ is updated with the computed $OC_{h_n}(t^+)$ from Eq. 2, which assist in determining the operability in future time instances. We can get $POC_{h_n}(t)$ from the design documents or initialize to 1.00 if not available otherwise.

Case II: Multiple Tasks to Single Host 'AND' Inter-dependency. As shown in Fig. 3(b1), tasks t_1 to t_n have 'AND' dependencies on host machine h_n with the strength of dependencies α_{1n} to α_{nn}. Figure 3(b2) depicts the relationships in the form of MIDG. We compute the operability of host h_n at the time instant t^+ (i.e., $OC_{h_n}(t^+)$) using $POC_{h_n}(t)$ and the 'AND' dependency of task impacts $\{I_{t_1}(t^+), \ldots, I_{t_1}(t^+)\}$. The combined impact factor due to the 'AND' dependency relations are as follows.

$$IF_{h_n}(t^+) := min(\alpha_{1n} I_{t_1}(t^+), \alpha_{2n} I_{t_2}(t^+), \ldots, \alpha_{nn} I_{t_n}(t^+)) \tag{4}$$

We utilize $IF_{h_n}(t^+)$ from Eq. 4 in Eq. 2 to compute the operability and in Eq. 3 to update the present operability.

Case III: Multiple Tasks to Single Host 'OR' Inter-dependency. As shown in Fig. 3(c1), tasks t_1 to t_n have an 'OR' dependency on host h_n with the strength of dependencies α_{1n} to α_{nn} respectively. Figure 3(c2) shows the

node relationships in the form of MIDG. Similar to previous cases, first we compute the combined impact factor $IF_{h_n}(t^+)$ due to the 'OR' dependency of task impacts $\{I_{t_1}(t^+), \ldots, I_{t_1}(t^+)\}$ as in Eq. 5. We then utilize Eqs. 2 and 3 to compute $OC_{h_n}(t^+)$ using the present operational capability of $POC_{h_n}(t)$.

$$IF_{h_n}(t^+) := max(\alpha_{1n} I_{t_1}(t^+), \alpha_{2n} I_{t_2}(t^+), \ldots, \alpha_{nn} I_{t_n}(t^+)) \tag{5}$$

Case IV: Single Task to Multiple Host 'AND' Inter-dependency. As in Fig. 3(d1) a task t_n has an 'AND' dependency on hosts h_1 to h_n with the strength of dependency α_{1n} to α_{nn}, respectively. Here, we can formulate the impact factor $IF_{h_n}(t^+)$ on host h_n, and operational capability $OC_{h_n}(t^+)$ of h_n as follows.

$$IF_{h_n}(t^+) := min(\alpha_{n1}, \alpha_{n2}, \ldots, \alpha_{nn}) I_{t_n}(t^+) \tag{6}$$

We then utilize Eqs. 2 and 3 to compute and update the operability.

Case V: Single Task to Multiple Hosts 'OR' Inter-dependency. Similar to previous case, we can formulate the impact factor $IF_{h_n}(t^+)$ on host h_n as follows.

$$IF_{h_n}(t^+) := max(\alpha_{n1}, \alpha_{n2}, \ldots, \alpha_{nn}) I_{t_n}(t^+) \tag{7}$$

Again, we then utilize Eqs. 2 and 3 to compute and update the operability.

Case VI: Single Task to Single Host 'FLOW' Inter-dependency and Task to Task Intra-dependency. As shown in Fig. 3(f1) and (f2), we have two tasks t_m, and t_n having 'FLOW' dependency on host h_m, and h_n. There is also an intra-dependency between tasks t_m and t_n with the strength of intra-dependency γ_{mn}, where t_m precedes t_n, meaning t_m needs to be completed first to perform t_n. Here, we compute the corresponding impact factors as below.

$$\left.\begin{aligned} IF_{h_m}(t^+) &:= \alpha_{mm} I_{t_m}(t^+) \\ IF_{h_n}(t^+) &:= \tfrac{1}{N}\left[\alpha_{nn} I_{t_n}(t^+) + \gamma_{mn} IF_{h_m}(t^+)\right] \end{aligned}\right\} \tag{8}$$

In Eq. 8, N is the total number of dependency of task t_n which includes both intra and inter-dependencies, which is equal to 2 in this case. If $\gamma_{mn} = 0$, means there is no intra-dependency, then $N = 1$, and the Eq. 8 becomes same as the 'FLOW' dependency as in Eq. 1. We then compute the operability of h_n and h_m as below.

$$\left.\begin{aligned} OC_{h_m}(t^+) &:= max(POC_{h_m}(t) - IF_{t_m}(t^+), 0) \\ OC_{h_n}(t^+) &:= max(POC_{h_n}(t) - IF_{t_n}(t^+), 0) \end{aligned}\right\} \tag{9}$$

Mission Impact and Overall Operational Capability. Let us consider, a mission m can be accomplished using a set of task paths $\{TP_1, TP_2, \ldots, TP_n\}$, where each task path TP_n itself includes a sequence of tasks $\{t_{n_1}, t_{n_2}, \ldots, t_{n_m}\}$ to perform to complete the task path TP_n. Also, the corresponding host paths $\{HP_1, HP_2, \ldots, HP_m\}$, where each host path HP_m itself includes a sequence of hosts $\{h_{n_1}, h_{n_2}, \ldots, h_{n_m}\}$. Then, we can compute the overall mission impact $I_m(t^+)$ and overall operational capability measure of the mission $OCM_m(t^+)$ using the below equation.

$$\left.\begin{aligned} I_m(t^+) &= \max_{k \in TP} \left(\frac{\sum_{j \in TP_k} IF_{kj}(t^+)}{|TP_j|} \right) \\ OCM_m(t^+) &= \min_{n \in HP} \left(\frac{\sum_{j \in HP_n} OC_{h_{nj}}(t^+)}{|HP_j|} \right) \end{aligned} \right\} \tag{10}$$

Here, $|TP_j|$ is the cardinality of the task path TP_j, $IF_{kj}(t^+)$ is the impact factor of task j in task path k. Similarly, $|HP_j|$ is the cardinality of the host path HP_j. $OC_{h_{nj}}(t^+)$ is the operational capability of dependent host j (associated with task j) in host path path n (associated with task path k) at time instance t^+. The interpretation of Eq. 10 is that we take the maximum of the average of all impact factors in the task paths for mission impact computations. We take the minimum of the average of all operability measures for the corresponding host paths to compute system operability due to mission impact.

5 Case Study Formulation and Evaluation

We have set up a case study of SCADA operations in EDS, as shown in Fig. 4. The mission here is to *maintaining proper functioning of the SCADA operations*. Here, we have eight tasks associated with this mission. Task T_1 is to sense the field-device data by the sensors. T_2 is the I/O communications to the sensors. T_3 is to collate information from the sensors and I/O modules and provide them to the control screen in HMI. T_4 presents the real-time graphical schematic/mimic diagrams of the plant operations. T_5 accumulates the time-series historical data. T_6 is the alarm and event log monitoring using real-time and historical data. T_7 is to manually monitor the operational levels of the field-devices by the operators using appropriate CLIs (command-line interface). Finally, T_8 is to initiate supervisory commands to set the operational levels if necessary. Because of redundant operational logic and intra-dependencies of the tasks, the mission can be accomplished by the sequence of tasks $T_3 \rightarrow T_6 \rightarrow T_8$ (i.e., task path1), or $T_3 \rightarrow T_5 \rightarrow T_7 \rightarrow T_8$ (i.e., task path2), where $T_3 := T_1 \wedge T_2$, and $T_6 := T_4 \wedge T_5$. Here, $\wedge$ means logical 'AND'. By task path, we mean the sequence of tasks needed to perform to have the mission successful.

Mission Dependency for the Case Study. The tasks that we have mentioned in Fig. 4 depend on the programs or applications running on the underlying hosts or network devices. Here, we only consider the task-host interdependency bypassing the task-application-host dependency using transitivity rule. For example, the job T_1 of sensing relies on the sensors, T_2 relies on the I/O modules (RTU/IED). T_3 depends on both T_1 and T_2. Similarly, we present other task-host inter-dependencies and task-task intra-dependencies associated with the mission in Fig. 4. The associated network devices are h_1 to h_8, as in Fig. 4.

5.1 Network Architecture and Possible Exploitation Scenarios

We have set up a six subnet EDS network based on NIST recommended defense-in-depth architecture [3] and following the standards IEC 60870-5 (SCADA) in Fig. 5. Here, we have *webServer*, and *workStation*1 in the corporate network. The *webServer* allows *http* and *https* traffic from the internet. In the control layer, we have *controlFirewall*, *historian*, and *workStation*2. *historian* allows *ftp* from *workStation*1. *workStation*2 allows *ssh* from *workStation*1. The supervisory computers $WS3$ collates information from $DataAcquisitionServer(DAS)$ which itself collects data from the *sensors* and $PLC/RTU/IED$ devices. The *historian* can do *ftp* to DAS; *historian* allows pulling data from *workStation*2 as they are on the same subnet. Also, $WS3$

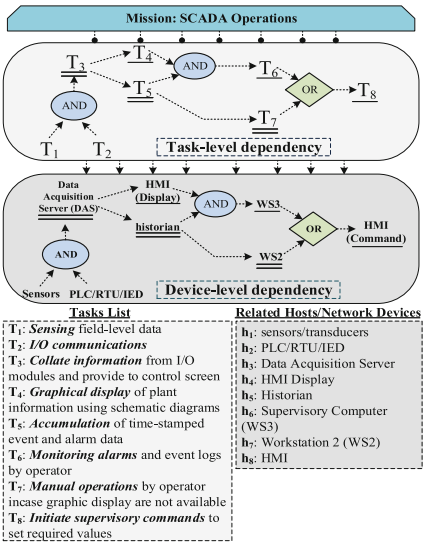


Fig. 4. Mission dependency for the case study. Double underlines indicate direct impact because of the vulnerabilities in associated devices; single underline shows an indirect impact

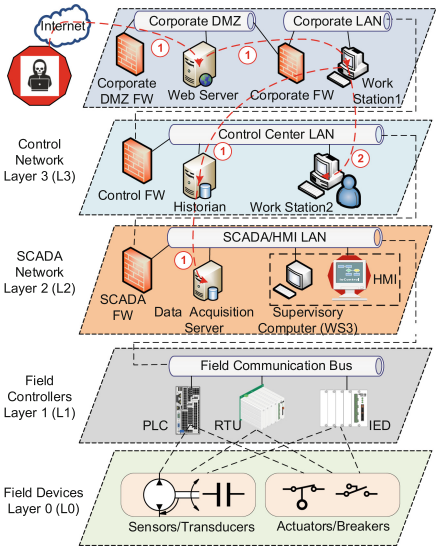


Fig. 5. NIST defense-in-depth architecture based network setup for the case study

Table 1. Network vulnerability information

Vulnerability	CVSS score	Associated host	Exploitation result
CVE-2020-5847[a]	9.8	webServer	Remote code execution
CVE-2019-18822[b]	5.9	workStation1	Privilege escalation
CVE-2020-0796[c]	10.0	workStation2	Remote code execution
CVE-2019-11013[d]	6.5	Historian	Directory traversal
CVE-2020-1008[e]	8.8	DataAcquisitionServer	Remote code execution

[a] https://nvd.nist.gov/vuln/detail/CVE-2020-5847
[b] https://nvd.nist.gov/vuln/detail/CVE-2019-18822
[c] https://nvd.nist.gov/vuln/detail/CVE-2020-0796
[d] https://nvd.nist.gov/vuln/detail/CVE-2019-11013
[e] https://nvd.nist.gov/vuln/detail/CVE-2020-1008

directly connects with the HMI. *workStation*2 houses programs/CLI applications for direct monitoring of PLC/- RTU/IED devices. From the HMI, it is possible to initiate corrective commands to set/adjust the operational levels of some physical devices which pass to the actuators.

Irrespective of the network administrators efforts to make the network free of any exploitation points, there are a couple of vulnerabilities existing on the network, as shown in the Table 1. At first, the attacker can exploit CVE-2020-5847 of the *webServer*1 to by executing codes remotely and gain privilege in corporate LAN. Then, the attacker can exploit another *privilege escalation* vulnerability CVE-2019-18822 in *workStation*1 to gain access to the control LAN. From, *workStation*1 the attacker can exploit CVE-2019-11013 in the *historian*, which allows directory traversal. Also, the attacker may reach to *workstation*2 through exploiting CVE-2020-0796 and gain *remote code execution privilege* in control LAN. Again, from the *historian*, the attacker can exploit *DataAcquisitionServer* by exploiting the vulnerability CVE-2020-1008. Thus, the attacker can utilize two different attack paths, marked as ① and ②. The attacker can not reach HMI or $WS3$ because these devices don't have any vulnerabilities. Thus, the attacker can directly impact tasks T_3, T_5, T_7, as shown by double underline marks in Fig. 4. The impact on these three tasks impacts tasks T_4, T_6, T_8 indirectly (as shown by single underline mark) because of the dependency relationships. Either way, the attacker's goal is to disrupt the SCADA operations, and our mission is to secure the same.

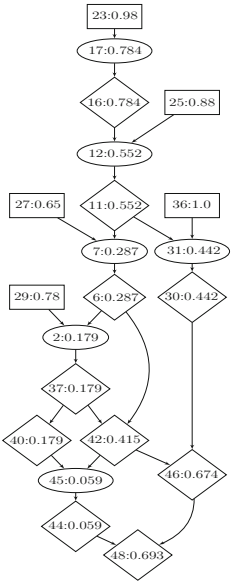


Fig. 6. Pruned mission impact assessment graph for the case study (nodes Datalog clauses are given in Table 2)

Table 2. Datalog clauses of Fig. 6

Node	Datalog clauses
23	vulExists(webServer,'CVE-2020-5847',httpd,remoteExploit,privEscalation)
17	RULE 2 (remote exploit of a server program)
16	execCode(webServer,root)
25	vulExists(workStation1,'CVE-2019-18822',sshd,remoteExploit,privEscalation)
12	RULE 2 (remote exploit of a server program)
11	execCode(workStation1,root)
36	vulExists(workStation2,'CVE-2020-0796',sshd,remoteExploit,privEscalation)
27	vulExists(historian,'CVE-2019-11013',ftpd,remoteExploit,privEscalation)
7	RULE 2 (remote exploit of a server program)
6	execCode(historian,root)
31	RULE 3 (local exploit of a server program)
30	execCode(workStation2,root)
29	vulExists(dataAcquisitionServer,'CVE-2020-1008',ftpd,remoteExploit,privEscalation)
2	RULE 3 (local exploit of a server program)
37	taskImpact(t3CollateFieldData)
40	taskImpact(t4GraphicDisplay)
42	taskImpact(t5AccumulateEventLogs)
45	RULE 35 (An and-dependent task impacts later task)
44	taskImpact(t6MonitoringAlarms)
46	taskImpact(t7ManualStatusCheck)
48	taskImpact(t8InitiateCorrectiveCommands)

5.2 Result Analysis and Interpretations

We present the complete MIAG in Fig. 8 in Appendix because of space constraints. We utilize a pruning process on the MIAG as in Fig. 6, where we eliminate the condition nodes except the vulnerability nodes; we keep the privilege and task impact nodes, including the dependency rule nodes. On the pruned MIAG, we apply the score quantification process discussed in Subsect. 4.6 and 4.7.

Task Impact Computation: From, Fig. 6, we find the computed derived scores for the task impacts. Thus, $taskImpact(t1Sensing) = 0.0, taskImpact(t2IOCommunication) = 0.0, taskImpact(t3CollateFieldData) = 0.179, taskImpact(t4GraphicDisplay) = 0.179, taskImpact(t5AccumulateEventLogs) = 0.415, taskImpact(t6MonitoringAlarms) = 0.059, taskImpact(t7ManualStatusCheck) = 0.674, taskImpact(t8InitiateCorrectiveCommand) = 0.693$.

Task Impact Interpretation: There is no way the attacker can impact T_1 and T_2 because these devices have no vulnerability and exploitation path. Thus, both task impacts are zero. Although task T_3 'AND'-depends on T_1 and T_2, because of the vulnerability exploitation in $DataAcquistionServer$, the task T_3 can be impacted; thus the score is 0.179 as derived from the pruned MIAG graph. T_4 'Flow'-depends on T_3, and has no other associated vulnerability. Thus, the

impact on task T_4 is equal to the derived impact of task T_3, which is 0.179. Task T_5 'Flow'-depends on T_3 and also can be impacted by exploitation of *historian*. Thus, the derived score of T_5 is $(6 : execCode(historian, root)) \vee (37 : taskImpact(t3CollateFieldData)) = 0.287 \vee 0.179 = 0.287 + 0.179 - 0.287 * 0.179 = 0.415$. Similarly, we can compute the other derived scores for the task impacts on $T_6, T_7, T8$. From the pruned MIAG, we see that even if T_4, T_6, and T_8 don't have direct vulnerabilities on the associated host devices, but these tasks are getting impacted because of their dependency relations on other preceding tasks. The task impact score represents the fraction of disruption on the task on a scale of 0–1. The higher the score, the greater is the possible disruption on the task.

Overall Mission Impact and Operability Computation and Interpretation: We provide the computed impact factor and operability for the two task paths (i.e., the two ways the mission can be established) in Tables 3, 4, 5, and 6, for different combinations of the strength of dependency factors. Here, scenario ① means all $\alpha_{ii} = 1.0$, and $\gamma_{ij} = 1.0$. Scenario ②, ③, and ④ means all α_{ii} is equal to 0.75, 0.5, and 0.25, respectively keeping $\gamma_{ij} = 1.0$. Finally, we compute the mission impact and overall system operability using Eq. 10 and Tables 3, 4, 5, and 6. We have assumed $POC(h_i) = 1.0$ for this illustration, where $i = 1, 2, \ldots, 8$. For the scenario ① of equal inter-SOD and intra-SOD, the mission impact is $max(0.1382, 0.2305) = 0.2305$, and operational capability is $min(0.8618, 0.7695) = 0.7695$ utils. This means the SCADA operational mission would have 76.95% predictive operability having 23.05% impacts (i.e., disruptions) posed by the existing vulnerabilities during a cyberattack incident.

Table 3. Impact factor for task path1: $T_3 \rightarrow T_6 \rightarrow T_8$ ($T_3 = T_1 \wedge T_2, T_6 = T_4 \wedge T_5$)

Scenario	Impact Factor (IF)							
	IF_{h1}	IF_{h2}	IF_{h3}	IF_{h4}	IF_{h5}	IF_{h6}	IF_{h8}	Average
①	0.00	0.00	0.0895	0.1343	0.2523	0.0966	0.4011	0.1382
②	0.00	0.00	0.0671	0.1007	0.1892	0.0725	0.3008	0.1037
③	0.00	0.00	0.0448	0.0671	0.1261	0.0483	0.2005	0.0691
④	0.00	0.00	0.0224	0.0336	0.0631	0.0242	0.1003	0.0346

Table 4. Impact factor for task path2: $T_3 \rightarrow T_5 \rightarrow T_7 \rightarrow T_8$ ($T_3 = T_1 \wedge T_2$)

Scenario	Impact Factor (IF)						
	IF_{h1}	IF_{h2}	IF_{h3}	IF_{h5}	IF_{h7}	IF_{h8}	Average
①	0.00	0.00	0.0895	0.2523	0.4631	0.5781	0.2305
②	0.00	0.00	0.0671	0.1892	0.3473	0.4335	0.1729
③	0.00	0.00	0.0448	0.1261	0.2316	0.2890	0.1152
④	0.00	0.00	0.0224	0.0631	0.1158	0.1445	0.0576

Table 5. Host operability for task path1: $T_3 \rightarrow T_6 \rightarrow T_8$ ($T_3 = T_1 \wedge T_2, T_6 = T_4 \wedge T_5$), associated hosts $= \{h_1, h_2, h_3, h_4, h_5, h_6, h_8\}$

Scenario	Operational Capability (OC)							
	OC_{h1}	OC_{h2}	OC_{h3}	OC_{h4}	OC_{h5}	OC_{h6}	OC_{h8}	Average
①	1.00	1.00	0.9105	0.8658	0.7478	0.9034	0.6052	0.8618
②	1.00	1.00	0.9329	0.8993	0.8108	0.9275	0.7039	0.8963
③	1.00	1.00	0.9553	0.9329	0.8739	0.9517	0.8026	0.9309
④	1.00	1.00	0.9776	0.9664	0.9369	0.9758	0.9013	0.9654

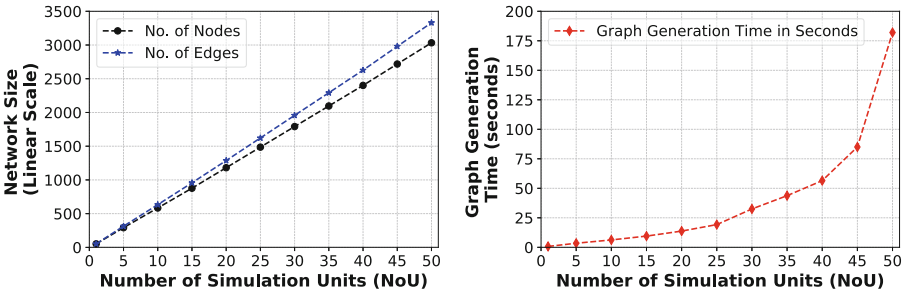
Table 6. Host operability for task path2: $T_3 \rightarrow T_5 \rightarrow T_7 \rightarrow T_8$ ($T_3 = T_1 \wedge T_2$), associated hosts $= \{h_1, h_2, h_3, h_5, h_7, h_8\}$

Scenario	Operational Capability (OC)						
	OC_{h1}	OC_{h2}	OC_{h3}	OC_{h5}	OC_{h7}	OC_{h8}	Average
①	1.00	1.00	0.9105	0.7478	0.5369	0.4219	0.7695
②	1.00	1.00	0.9329	0.8108	0.6527	0.5665	0.8271
③	1.00	1.00	0.9553	0.8739	0.7684	0.7110	0.8848
④	1.00	1.00	0.9776	0.9369	0.8842	0.8555	0.9424

5.3 Evaluation of Scalability

In our mission impact model, generating the mission impact assessment graph takes time depending on the network size and, thus, related to the scalability. Therefore, we present here a comparison of the simulation time to generate the MIAG. We consider the full SCADA case study as one unit and then replicate and grouped the units together (i.e., no. of simulation units termed as 'NoU').

We utilize an Ubuntu 18.04 OS version Virtual Box (VB) image with 50 GB hard drive and 8 GB memory to install MulVAL and perform the simulation. We use the VB in a Dell Laptop with Windows 10 OS, 16 GB RAM, and 1 TB hard disk. We present the number of nodes and edges and the average graph generation time in Fig. 7. With the increase of simulation units, the graph generation time increases exponentially. Still, as we can see, it can generate the graph with nearly 3000 nodes and 3300 edges in around 182 s (∼3 min) on average. Cao et al. [2] also present a detailed evaluation of scalability using the MulVAL simulation platform. We encourage readers to follow the explanations presented by Cao et al. [2] in Subsect. 5.3 to get a good insight on scalability using MulVAL.



(a) (b)

Fig. 7. Evaluation of scalability: (a) Network size in terms of number of nodes and edges, (b) Graph generation time (in sec.) vs. number of simulation units

5.4 Generalization Challenges and Way Forward

We demonstrate the model for energy systems. The model applies to other cyber-physical systems as well. There may be some challenges in identifying the dependencies and determining the strength of dependencies. We can overcome these utilizing artificial intelligence techniques and Hidden-Markov models (HMM) using system process calls. We may also utilize FDNA [4] for dependency modeling. There can be issues with cyclic graphs, and the MulVAL engine is capable of handling the cycling problems [6]. Choosing the right tool to simulate large-scale CPS is also challenging. Open-source tools such as MulVAL, NetworkX can help in this regard. The most critical challenge would be to incorporate real-time services as comprehensive as possible. Rather than considering the complete system as a single mission, dividing the whole mission into several sub-missions and performing mission-critical dependency modeling and impact assessment for each subdivided mission could provide some guidance. The creation of customized rule sets based on the dependency scenario is another challenge to take into considerations.

6 Related Work

Gabriel [8] presents mission security situation assessment using impact dependency graphs. The article presents a conceptual framework for the cyberattack model and uses logical constraint graphs to assess the assets' operational capacity and mission impact. Albanese and Jajodia [10] present a graphical model to evaluate the effects of multi-step attacks. The authors propose an impact assessment graph utilizing the vulnerability dependency graph. Changwei et al. [11] model intrusion evidence dependency using probabilistic evidence graph and attack graph. The use of qualitative evaluation questions the applicability of the model in practical settings. Jajodia et al. [12] present 'Cauldron,' a mission-centric cyber situational awareness tool. Cauldron focuses more on the implementation side and does not disclose the underlying modeling details because of the nature

of operations. Haque et al. [13,14] utilize graph theory in analyzing impact and security in cyber-physical systems.

Sun et al. [1,15] propose a technique to model the dependency relations by utilizing the mission or service dependency graph and attack graph. Cao et al. [2] present a quantitative metric for business process impact assessment using the attack graphs and entity dependency graph. The works offer some directions on integrating system dependency in the security models where we get our primary motivations for this work. However, the articles focus on Enterprise networks and lack in providing formal mathematical reasoning to integrate the attack graph and dependency graph. Also, the correlation of the impact with system operability is missing. We have clearly defined different graph types and the integration process to use irrespective of the system type and the simulation platform. Although we have used the MulVAL simulation platform, our detailed mathematical formulations illustrate that the model is implementable regardless of the simulation platform. The significant difference between our model and the previous models is that we relate the impact to system operability. This would guide in developing mitigation strategies and defensive actions. The model is also applicable to the impact assessment of other cyber-physical systems with necessary adjustments.

7 Conclusion and Future Directions

This work addresses two existing modeling problems in the security domain for the energy delivery systems. The first problem deals with the incorporation of the mission functional dependency into the traditional attack graph model. The second problem is to quantify and assess the mission impact and system operability. We provide detailed formal explanations of the graphical models that we use to address the above problems. We offer a case study of SCADA operations for energy delivery systems and present the details of how the mathematical models can help assess the mission impact and system operability in the presence of vulnerabilities on the mission-critical devices. We plan to optimize our model by including control measures that would assist in minimizing the mission impact and maximizing the system operability to provide cyber resilience guidance. We only consider operational dependencies in this work. We have the plan to incorporate strategic and tactical dependencies in our future extension of this work.

Acknowledgment. This material is based upon work supported by the Department of Energy under Award Number DE-OE0000780.

A Appendix

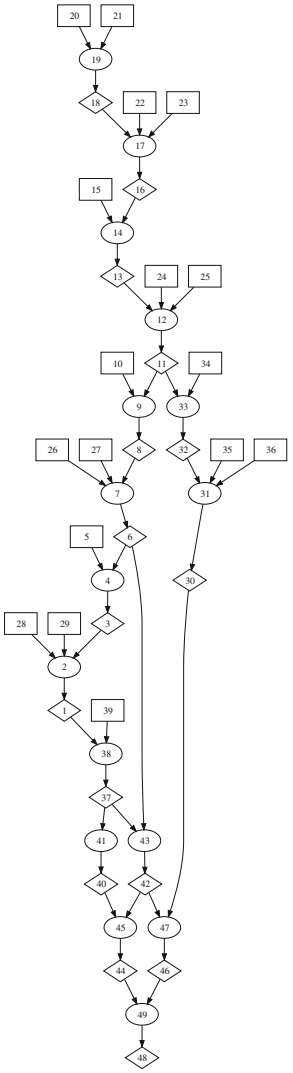


Fig. 8. Complete mission impact assessment graph for the case study (nodes Datalog clauses are given in Table 7)

Table 7. Datalog clauses of Fig. 8

Node	Datalog clauses
20	hacl(internet,webServer,httpProtocol,httpPort)
21	attackerLocated(internet)
19	RULE 11 (direct network access)
18	netAccess(webServer,httpProtocol,httpPort)
22	networkServiceInfo(webServer,httpd,httpProtocol,httpPort,root)
23	vulExists(webServer,'CVE-2020-5847',httpd,remoteExploit,privEscalation)
17	RULE 2 (remote exploit of a server program)
15	hacl(webServer,workStation1,sshProtocol,sshPort)
16	execCode(webServer,root)
14	RULE 10 (multi-hop access)
13	netAccess(workStation1,sshProtocol,sshPort)
24	networkServiceInfo(workStation1,sshd,sshProtocol,sshPort,root)
25	vulExists(workStation1,'CVE-2019-18822',sshd,remoteExploit,privEscalation)
12	RULE 2 (remote exploit of a server program)
10	hacl(workStation1,historian,ftpProtocol,ftpPort)
11	execCode(workStation1,root)
34	hacl(workStation1,workStation2,sshProtocol,sshPort)
9	RULE 10 (multi-hop access)
33	RULE 10 (multi-hop access)
26	networkServiceInfo(historian,ftpd,ftpProtocol,ftpPort,root)
27	vulExists(historian,'CVE-2019-11013',ftpd,remoteExploit,privEscalation)
8	netAccess(historian,ftpProtocol,ftpPort)
32	netAccess(workStation2,sshProtocol,sshPort)
35	networkServiceInfo(workStation2,sshd,sshProtocol, sshPort,root)
36	vulExists(workStation2,'CVE-2020-0796',sshd,remoteExploit,privEscalation)
7	RULE 2 (remote exploit of a server program)
31	RULE 2 (remote exploit of a server program)
5	hacl(historian,dataAcquisitionServer,ftpProtocol,ftpPort)
6	execCode(historian,root)
4	RULE 10 (multi-hop access)
30	execCode(workStation2,root)
28	networkServiceInfo(dataAcquisitionServer,ftpd,ftpProtocol, ftpPort,root)
29	vulExists(dataAcquisitionServer,'CVE-2020-1008',ftpd,remoteExploit,privEscalation)
3	netAccess(dataAcquisitionServer,ftpProtocol,ftpPort)
2	RULE 2 (remote exploit of a server program)
1	execCode(dataAcquisitionServer,root)
39	task(t3CollateFieldData,dependency,dataAcquisitionServer)
38	RULE 32 (An impacted host impacts dependent task)
37	taskImpact(t3CollateFieldData)
41	RULE 33 (An impacted task impacts Flow-dependent succeeding task)
43	RULE 34 (An impacted task or an impacted host impacts dependent task)
40	taskImpact(t4GraphicDisplay)
42	taskImpact(t5AccumulateEventLogs)
45	RULE 35 (An and-dependent task impacts following task)
47	RULE 36 (An impacted task or an impacted host impacts dependent task)
44	taskImpact(t6MonitoringAlarms)
46	taskImpact(t7ManualStatusCheck)
49	RULE 37 (Both impacted tasks impacts dependent task)
48	taskImpact(t8InitiateCorrectiveCommands)

Note: Node numbers are in the order from top to bottom as in Fig. 8

References

1. Sun, X., Liu, P., Singhal, A.: Toward cyberresiliency in the context of cloud computing [resilient security]. IEEE Secur. Priv. **16**(6), 71–75 (2018)
2. Cao, C., Yuan, L.-P., Singhal, A., Liu, P., Sun, X., Zhu, S.: Assessing attack impact on business processes by interconnecting attack graphs and entity dependency graphs. In: Kerschbaum, F., Paraboschi, S. (eds.) DBSec 2018. LNCS, vol. 10980, pp. 330–348. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95729-6_21
3. Stouffer, K., Falco, J.: Recommended practice: improving industrial control systems cybersecurity with defense-in-depth strategies. Department of Homeland Security, Control systems security program, National cyber security division (2009)
4. Garvey, P.R., Pinto, C.A.: Introduction to functional dependency network analysis. In: The MITRE Corporation and Old Dominion, 2nd International Symposium on Engineering Systems, MIT, Cambridge, Massachusetts, vol. 5 (2009)
5. Ou, X., Govindavajhala, S., Appel, A.W.: MulVAL: a logic-based network security analyzer. In: USENIX Security Symposium, Baltimore, MD, vol. 8, pp. 113–128 (2005)
6. Gonda, T., Pascal, T., Puzis, R., Shani, G., Shapira, B.: Analysis of attack graph representations for ranking vulnerability fixes. In: GCAI, pp. 215–228 (2018)
7. Rao, B., Mitra, A.: An approach to merging of two community subgraphs to form a community graph using graph mining techniques. In: 2014 IEEE International Conference on Computational Intelligence and Computing Research, pp. 1–7. IEEE (2014)
8. Jakobson, G.: Mission-centricity in cyber security: architecting cyber attack resilient missions. In: 2013 5th International Conference on Cyber Conflict, CYCON 2013, pp. 1–18. IEEE (2013)
9. Guariniello, C., DeLaurentis, D.: Supporting design via the system operational dependency analysis methodology. Res. Eng. Des. **28**(1), 53–69 (2017)
10. Albanese, M., Jajodia, S.: A graphical model to assess the impact of multi-step attacks. J. Def. Model. Simul. **15**(1), 79–93 (2018)
11. Liu, C., Singhal, A., Wijesekera, D.: Mapping evidence graphs to attack graphs. In: IEEE International Workshop on Information Forensics and Security, WIFS 2012, pp. 121–126. IEEE (2012)
12. Jajodia, S., Noel, S., Kalapa, P., Albanese, M., Williams, J.: Cauldron mission-centric cyber situational awareness with defense in depth. In: 2011 Military Communications Conference, MILCOM 2011, pp. 1339–1344. IEEE (2011)
13. Haque, M.A., Shetty, S., Krishnappa, B.: Modeling cyber resilience for energy delivery systems using critical system functionality. In: Resilience Week, RWS 2019, vol. 1, pp. 33–41. IEEE (2019)
14. Haque, M.A., Shetty, S., Krishnappa, B.: Cyber-physical system resilience: frameworks, metrics, complexities, challenges, and future directions. In: Complexity Challenges in Cyber Physical Systems: Using Modeling and Simulation (M&S) to Support Intelligence, Adaptation and Autonomy, pp. 301–337 (2019)
15. Sun, X., Singhal, A., Liu, P.: Towards actionable mission impact assessment in the context of cloud computing. In: Livraga, G., Zhu, S. (eds.) DBSec 2017. LNCS, vol. 10359, pp. 259–274. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61176-1_14