energies

IMPACT FACTOR 3.0

CITESCORE 6.2

Article

SecuriDN: A Modeling Tool Supporting the Early Detection of Cyberattacks to Smart Energy Systems

Davide Cerotti, Daniele Codetta Raiteri, Giovanna Dondossola, Lavinia Egidi, Giuliana Franceschinis, Luigi Portinale, Davide Savarro and Roberta Terruggia

Special Issue
Model Predictive Control-Based Approach for Microgrids

Edited by
Dr. Valerio Mariani, Prof. Dr. Giorgio Graditi, Dr. Maria Valenti and Prof. Dr. Luigi Glielmo

MDPI

https://doi.org/10.3390/en17163882

energies

MDPI

Article

SecuriDN: A Modeling Tool Supporting the Early Detection of Cyberattacks to Smart Energy Systems

Davide Cerotti [1,2], Daniele Codetta Raiteri [1,2], Giovanna Dondossola [3], Lavinia Egidi [1], Giuliana Franceschinis [1,2,*], Luigi Portinale [1,2], Davide Savarro [4] and Roberta Terruggia [3]

[1] Computer Science Institute, DiSIT, Università del Piemonte Orientale (UPO), 15121 Alessandria, Italy; davide.cerotti@uniupo.it (D.C.); daniele.codetta@uniupo.it (D.C.R.); lavinia.egidi@uniupo.it (L.E.); luigi.portinale@uniupo.it (L.P.)
[2] Consorzio Nazionale Interuniversitario per le Telecomunicazioni (CNIT), 43124 Parma, Italy
[3] Transmission and Distribution Technologies Department, Ricerca sul Sistema Energetico (RSE S.p.A.), 20134 Milano, Italy; giovanna.dondossola@rse-web.it (G.D.); roberta.terruggia@rse-web.it (R.T.)
[4] Computer Science Department, Università di Torino, 10149 Torino, Italy; davide.savarro@unito.it
* Correspondence: giuliana.franceschinis@uniupo.it; Tel.: +39-0131-360159

Abstract: SecuriDN v. 0.1 is a tool for the representation of the assets composing the IT and the OT subsystems of Distributed Energy Resources (DERs) control networks and the possible cyberattacks that can threaten them. It is part of a platform that allows the evaluation of the security risks of DER control systems. SecuriDN is a multi-formalism tool, meaning that it manages several types of models: architecture graph, attack graphs and Dynamic Bayesian Networks (DBNs). In particular, each asset in the architecture is characterized by an attack graph showing the combinations of attack techniques that may affect the asset. By merging the attack graphs according to the asset associations in the architecture, a DBN is generated. Then, the evidence-based and time-driven probabilistic analysis of the DBN permits the quantification of the system security level. Indeed, the DBN probabilistic graphical model can be analyzed through inference algorithms, suitable for forward and backward assessment of the system's belief state. In this paper, the features and the main goals of SecuriDN are described and illustrated through a simplified but realistic case study.

Keywords: cyberattack detection; cyber physical power systems; distributed energy resources; Bayesian Networks; risk assessment; attack graphs; MITRE ATT&CK framework; IEC 61850; evidence-based and time-driven probabilistic analysis; multiformalism models

check for updates

Citation: Cerotti, D.; Codetta Raiteri, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Savarro, D.; Terruggia, R. SecuriDN: A Modeling Tool Supporting the Early Detection of Cyberattacks to Smart Energy Systems. *Energies* **2024**, *17*, 3882. https://doi.org/10.3390/en17163882

Academic Editor: Keke Gai

Received: 8 June 2024
Revised: 19 July 2024
Accepted: 1 August 2024
Published: 6 August 2024

1. Introduction

In the last decades, *electro-energetic infrastructures* have rapidly evolved from a unidirectional "producer to consumer" model to a more complex scenario in which the distributed resources can be both producers and consumers. The management of the power fluxes in this heterogeneous grid environment entails an increased exchange of information, and hence it requires high connectivity of the devices and control logic; as a consequence, the surface exposed to cyberattacks is extensive, making the whole system potentially vulnerable to adversarial activity. Cyberattacks to such critical systems may have severe consequences and put at risk national stability; for this reason, the issue is receiving great attention.

Several European legislative acts address the data exchanges of energy infrastructures and their cybersecurity. Within the electrical sector regulation, the EU Regulation 2017/1485 System Operation Guideline provides a set of indications including those concerning operation security for the transmission grid, consistent rules for transmission and distribution system operators and the Significant Grid User (SGU), concerning grid connection operations. The EU cybersecurity Directive NIS2, which became effective in 2023, updates the previous NIS Directive introduced in 2016, adding new cybersecurity obligations for essential and important critical infrastructure operators including, e.g., the operators of renewable power plants and electric vehicle charging infrastructures. Specifically, Article 21 of the NIS2 Directive states that "Member States shall ensure that essential and important entities take appropriate and proportionate technical, operational and organisational measures to manage the risks posed to the security of network and information systems [. . .] When assessing the proportionality of those measures, due account shall be taken of the degree of the entity's exposure to risks, the entity's size and the likelihood of occurrence of incidents and their severity, including their societal and economic impact". Each member state shall implement the NIS2 Directive by means of a series of national legislative acts. To comply with sector-specific and cybersecurity legislations, some national norms of the energy sector already include cybersecurity obligations based on international standards, e.g., the Norms CEI 0-16 [1] (high- and medium-voltage SGU connection rules) and CEI 0-21 [2] (low-voltage SGU connection rules). One of the reference standards in the Norm CEI 0-16 is the ISA/IEC 62443 [3] that addresses the support to incident management with the foundational requirement FR 6 "Timely response to events", a specific focus in our work.

In this obligation framework, tools are needed to help the cybersecurity analyst in facing the cybersecurity threats in a more informed way: indeed, automatic tools may help specialized personnel in the assessment of the potential risks and the evaluation of the appropriate countermeasures. Since cyberattacks progress through various steps of a cyber kill chain, the main objective is to enable the early detection of threats in order to discover adversarial activity in its first phases and possibly enact mitigation actions, to prevent any damage to the assets and any degradation of functionality. For this reason, we have developed a modular platform whose aim is to offer methodological and practical support to the cybersecurity analysts in evaluating the risk, both in the design phase, when the countermeasures placement is planned, but also in the monitoring phase, when the system is up and running, for a model based on early detection of ongoing attack processes. In this paper, we present SecuriDN, which is one component of such a platform.

SecuriDN includes a Graphical User Interface (GUI) and a few *solvers*. The GUI allows us to describe the configuration of the system under study and the possible attacks that may be directed towards the various assets; the solvers automatically derive a global attack graph (AG) and a Dynamic Bayesian Network (DBN) from such a description. The latter can then be exploited for a what-if analysis of possible attack scenarios and for online monitoring.

This paper is an extended and enriched version of [4]. With respect to [4], more details are provided here on the SecuriDN tool, and a realistic case study (of limited size for space limit reasons) is presented, considering the connection and the associated information exchanges of Distributed Energy Resources (DERs) to the power grid. The case study allows us to explain the features of SecuriDN and give a flavor of the whole process, from the design of the system structure, the modeling of its vulnerabilities and of the attack processes that may exploit it, to the evaluation of the potential risks.

Let us briefly describe the typical security assessment process exploiting our platform: the security analyst uses the SecuriDN GUI to input a representation of the data networks, with their hardware and software assets, and the communication channels connecting them. Such components come from a predefined and customisable library, and embed relevant information on the possible attacks that may be directed to each of them, as well as the way an attack process may propagate within the network. The GUI allows the analysts to switch between the system architecture level and the attack model level, enabling them to edit both the external and internal topology. Once the architecture and local attack descriptions are complete, SecuriDN generates a global AG and a corresponding DBN, which can then be used to compute the probability that a specific target is or will be compromised. The DBN can be used both to perform offline what-if analysis (e.g., comparing different configurations of the countermeasures, or different probabilistic characterizations of the attack steps) or as an online AI-based detection model when integrated in the framework's testbed for Intrusion Detection experimentation. Observe that the manual generation of the

AG and of the DBN would be time-consuming and error-prone, and hence its automatic production through SecuriDN is valuable support to the work of the security analyst.

SecuriDN is a prototype developed on top of DrawNET and the DNlib, and leverages the ability of DrawNET v. 3.7 to deal with multiple user-defined graph-based formalisms. Moreover, for the specification of the possible attack steps and their composition, Meta-Attack Language (MAL) [5] has been considered and adapted.

The SecuriDN tool, which is the focus of this paper, is part of a comprehensive platform [6] designed to create a flexible detection system for adversarial activities. It operates on the cybersecurity analyst's workstation, allowing the development of new models to analyze evidence of attack steps. These models are deployed as detection modules within the platform. Evidence gathered from the monitored network is filtered in real time through an OpenSearch database pipeline and delivered to the detection modules via a communication channel based on a producer–consumer model. This setup allows multiple detection modules to coexist and collaborate simultaneously. The analysis results from these modules are then displayed to the analyst through the OpenSearch Dashboard, facilitating the detection of adversarial activity as it occurs. The platform can also be used as a testbed for the validation of detection models thanks to an emulated network and attacker. We have taken advantage of this functionality for our experiments in Section 6.

The paper is organised as follows: in Section 2, the related work is discussed; in Section 3, the relevant background on both the application domain and the adopted models is introduced; in Section 4, the platform including SecuriDN is briefly introduced; in Section 5, the structure and features of SecuriDN are described; and, in Section 6, the case study is presented, modeled through SecuriDN and analyzed by the DBN solver.

2. Related Work

In the last decade, we have witnessed the digital evolution of energy systems, making them more connected, smart, resilient and efficient. This transformation has had an impact on the entire energy supply chain from bulk generation, transmission and distribution to local generation and consumption. It requires new functionalities and infrastructures, and this poses new challenges in terms of Information and Communication Technologies (ICTs) and cybersecurity. New paradigms as virtualization and containerization represent solutions to reduce costs and make component maintenance and updates easier by means of resources traditionally tied to specific hardware. In energy control, the virtual power plant concept and ICT virtualisation techniques allow us to monitor and control an increasing number of DERs [7–9]. To create a higher level of abstraction and achieve optimal control and coordination among a wide variety of components, a unified view of this diverse setting is proposed. As noted in [9], to really implement such a solution, the use of virtualization and containerization techniques is useful to ensure the necessary hardware independence and facilitate the automated deployment throughout the entire infrastructure. This emerging trend requires us to address also attack techniques exploiting vulnerabilities exposed by, for instance, Docker [10] containerization technology (see Section 6.3 for further details).

Among the noteworthy security assessment frameworks, ADVISE (ADversary VIew Security Evaluation) [11,12] allows us to design an Attack Execution Graph (AEG), taking into account the possible attack goals and the attack steps to reach them, as well as the capabilities in terms of the access, skills and knowledge required for the attacker to attempt an attack step in the graph. Given the characteristics of specific adversaries, including also their propensity to incur costs, to spend time and to accept the risk of being detected, a State LookAhead Tree (SLAT) is generated, and simulation is used to explore the paths that the attacker may pursue to gather measures that provide a quantitative assessment of the possible consequences of adversarial activities. An interesting extension of this formalism is the Advise Meta Modeling Framework [13,14], a higher level structure that facilitates the creation of ADVISE models, starting from Meta-Model definitions that include templates and structures for defining the various elements within a security model (e.g., assets, attack steps and defenses). While

ADVISE, compared with our solution, offers a more extensive attacker characterization with the possibility to modify its behavior based on some predefined parameters, it does not provide a library of ready-to-use components (e.g., assets or attack steps) to facilitate the model design, which has to be performed from scratch for every specific use case.

Another important solution in the panorama of security assessment models is Securi-CAD [15]. This tool, developed by Foreseeti (a KTH Royal Institute of Technology spin-off company that has recently been acquired by Google and whose functionality is going to be included in Google Cloud Security Command Center [16]), allows us to integrate a model of the ICT (and OT) infrastructure and a description of possible attack steps targeting the assets of such an infrastructure. Starting from the MAL [5] specification, a domain-specific language can be defined. It can be used to describe the possible attack steps, the possible countermeasures and the way attacks can propagate in the infrastructure towards a given goal. A specific scenario can then be built, and from it an AG is automatically derived; finally, the critical attack paths are calculated, allowing us to evaluate the security posture of the system. In our platform, we adopted a MAL-derived approach for the architecture specification but, as a distinction, we integrated real-world analytics to evaluate how the attacks evolve over time (see Section 6.2 for further details).

In the context of AG generation, MulVAL [17] is a framework for conducting multi-host security analysis to assess potentially multi-stage cyberattacks. It is based on the Datalog programming language (a reduced version of Prolog), and can automatically generate a list of vulnerabilities present on each analyzed host. Aggregating this information with additional knowledge such as network configuration, user accounts on each host, the interdependencies between different vulnerabilities and the access policies permitted for each user, MulVAL can produce a complete AG of the examined infrastructure. While vulnerability identification is performed automatically through an extended version of the Open Vulnerability Assessment Language scanner, the remaining parts that are mandatory to perform the analysis must be defined by the security analyst using the Datalog language. This process can be nonintuitive and potentially prone to errors. In contrast, in SecuriDN, despite the possibly lower expressive power of the adopted formalism, the definition of assets within the architecture and their associations can be carried out by the analyst through the user interface, making it more accessible to non-expert users. Additionally, SecuriDN can generate a security assessment model capable of answering probabilistic queries based on the provided input evidence.

While SecuriDN is primarily focused on supporting security assessment tasks to determine the likelihood of a set of the attack steps leading to a potential power system instability, both references [18,19] are more focused on assessing the impact of cyberattacks on power flow stability, thus enabling risk assessment. Their effort to characterize attack interactions and their probabilities are less detailed and articulated compared with our approach, making our studies complementary in this regard.

In [18], a risk assessment framework for evaluating vulnerabilities and risks associated with cyber–physical systems in distribution grids was proposed. The solution introduces a DER model composed of three layers: the control layer, the physical system layer and the communication system layer. The control layer employs algorithms like Optimal Power Flow (OPF) and load-sharing control tailored to specific operational objectives. In the threat identification step, potential vulnerabilities are identified through the National Vulnerability Database (NVD) and the most common ones are modeled using analytical methods. In the final step, impact quantification is performed by combining the likelihood of an attack with its potential impact, measured in terms of financial repercussions and system stability. The effectiveness of the framework was validated using simulations on modified IEEE 13-node and 123-node test feeders, demonstrating its effectiveness in identifying vulnerabilities and assessing risks in distribution grids integrating DER. While this framework focuses on system responses to various attack scenarios separately, SecuriDN is a tool designed to support multi-stage cyberattack security assessment, allowing for the modeling of potentially more complex interdependencies within the cyber kill chain. Although their solution uses

Bayesian Networks (BNs) to compute the probabilities of attack success for each vulnerable node, it does not consider their temporal evolution like the DBNs generated by SecuriDN do.

In [19], a dynamic risk assessment model for Cyber Physical Power Systems (CPPSs) is proposed, focusing on assessing the network security vulnerabilities of SCADA systems in substations, as well as the physical consequences these vulnerabilities could have under malicious control. A key feature of the framework is the computation of the probability of successful exploitation for each vulnerability based on characteristics extracted from the Common Vulnerability Scoring System (CVSS), additional temporal information related to the disclosure time and an attribute characterizing the attacker's skills. Based on this pre-computed information, a Probabilistic Attribute attacking-path Graph is defined and the maximum likelihood path from the starting exploited vulnerability to the target attacker's goal is determined. The model also estimates the physical consequences of the cyberattack by calculating the minimum load shedding in N-1 contingencies, simulating the impact of a successful attack on the SCADA system that could lead to the tripping of transmission lines and generators, resulting in cascading failures and load shedding in the power system. Compared with this solution, SecuriDN uses a MAL-derived AG formalism that can represent more complex interdependencies between attack steps within the modeled infrastructure. While this framework in [19] considers only the best path leading the attacker to its predefined goal based on the pre-computed attach step probabilities, SecuriDN evaluates all the possible attack paths from the initial node to the target node, computing the probability of successful exploitation over time based on the estimated completion times and the interdependencies between attack steps.

CyberSAGE [20,21] is a tool that supports model-based evaluation of the security posture of an organization, providing the means for automating several tasks and decreasing the modeling effort required by the user (providing ready-to-use templates and libraries). It produces a Security Argument Graph combining a mal-activity scenario (a description of the attack workflow based on a subset of UML activity diagram elements), a system description (describing the network of system components, with properties and implemented defenses) and the adversary profile. A set of rules describes the logical relationships among these three levels, and a rule engine generates the Security Argument Graph. An algorithm computes the success probability of each attack step, and then of the whole attack workflow. The rules are an important part of the model description: reusable extension templates are defined to facilitate the user in preparing the necessary rule set. Similarly to our framework, this tool is capable of calculating completion probabilities of each attack step; however, it does not perform a time-dependent estimation, taking also into account the evidence, while our platform can provide this estimation thanks to the use of DBNs combined with security analytics.

As stated in [22,23], the need for effective dynamic risk assessment tools has led researchers to explore the use of BNs, where the possibility to easily modify prior failure probabilities and obtain updated results in real time can be a useful feature in the context of developing reactive solutions deployable in diverse environments. However, almost all the mentioned security assessment approaches do not derive the BN starting from a high-level model, more familiar to the security analyst, as we do with the algorithm detailed in Section 5.2.3. Instead, they are built from scratch for each specific use case. Additionally, none of these works explore the possibilities offered by DBNs, which, through the introduction of temporal dependencies [24], support the early detection and the fast counteraction to attacks in the online detection platform (see Section 3.5 for further details).

The approach described in [25], known as DETECT, focuses on an actual Intrusion Detection System (IDS), which is related to our proposal because it employs BNs derived from Attack Trees to identify evolving attacks. Events are collected and monitored within a specified time frame, and these observations are used to update the BN parameters. After each event-driven update, the model is resolved, potentially triggering an attack warning or alarm. Similarly, in our case, DBNs derived from AGs can be utilized for the online detection of cyberattacks.

In summary, the main novelty of our proposal is a tool that enables the security analyst to automatically produce a detection model from a user-defined graphical representation of the assets and their potential vulnerabilities: currently the SecuriDN tool allows us to draw such representation through a domain-specific and fairly intuitive GUI. This is part of a comprehensive detection platform that receives data analytics from the energy infrastructure.

3. Background

In this section, some definitions and key concepts are introduced, which are needed as a background knowledge basis for the next sections.

3.1. Industrial Control Systems and IT/OT Convergence

In Industrial Control Systems (ICSs), Operational Technology (OT) networks are the areas where programmable systems or devices interact with the physical environment. The digitisation of ICSs and, in particular, of the energy sectors comes through the integration of Information Technology (IT) systems with the OT ones. IT/OT convergence offers several benefits in terms of system availability and stability, and remote monitoring and control. Cybersecurity is an important challenge for an effective IT/OT convergence where emerging paradigms need to interact with legacy components. In this new landscape, different areas have to communicate and interact, each of them with different objectives and requirements in terms of performance and cybersecurity. Corporate networks can be indirectly connected with OT areas.

The reference architecture of the case study considered in this paper is inspired by the architecture of Figure 1. Although a more structured architecture with several protection layers between the corporate and the OT networks is typically implemented, we chose a simpler architecture for a more manageable but still realistic example. To move from the corporate network to the OT network, potential attackers will need to traverse a single DMZ (more levels of protection would require more lateral movements, exploiting different vulnerabilities), or even, as the 2015 and 2016 attacks to Ukraine grids show [26,27], often attackers manage to obtain user credentials to progress through networks. In this paper, we refer a simplified, but realistic, use case where the included components and configurations are taken from knowledge of the real energy networks and occurred attacks.

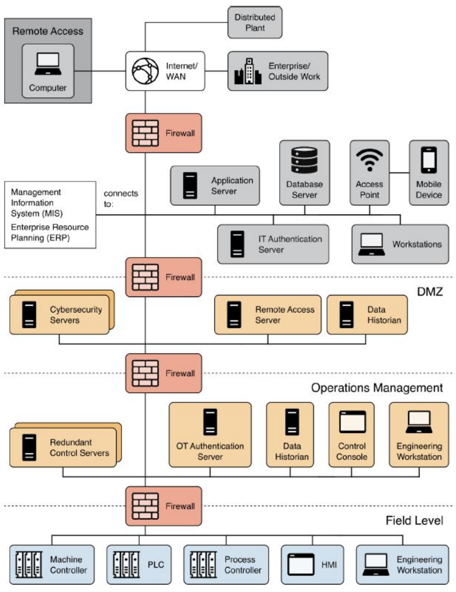

Figure 1. General ICS reference architecture [28].

3.2. IEC 61850

To address interoperability in CPPSs, several standards have been developed by international committees. The IEC 61850 [29] standard is increasingly used for the monitoring and control of distributed energy resources. This standard is structured in several parts, specifying both the underlying data model for the representation of energy entities and the communication protocols that can be used for exchanging those data. In particular, the Manufacturing Message Specification (MMS) protocol is used in substation–DER communications for the monitoring and control of DERs. The IEC 62351 standard series [30] specifies protocol-level security measures to protect the telecontrol communications based on protocols such as MMS. According to IEC 62351, secure MMS operates over Transport Layer Security (TLS), thus reducing the risk of cyberattacks such as Adversary-In-The-Middle (AITM) but not susceptibility to Distributed Denial of Service (DDoS) attacks.

3.3. MITRE ATT&CK Framework

To maintain a uniform terminology with the existing research and leverage the current knowledge about real-world cyberattacks, we chose to adopt the MITRE ATT&CK framework [31]. This comprehensive and constantly evolving repository about adversary tactics and techniques is widely used by researchers and organizations to systematically categorize, analyze and mitigate cyber threats. In our specific use case, this framework serves as an ontology that underpins the definitions of the attack steps included in our models. Some of these techniques are taken from the Enterprise Matrix [32], the most commonly used source focusing on attack methodologies related to desktop and cloud environments, and some are extracted from the ICS matrix [33], a specialized attack knowledge base for Industrial Control Systems, whose components are widely adopted in critical infrastructures.

The MITRE ATT&CK project is complemented by the MITRE Cyber Analytics Repository (CAR) [34], a knowledge base of analytics. It is compiled starting from the techniques of the ATT&CK matrix: for each technique, the data necessary to detect the adversarial step are identified and an appropriate sensor is implemented.

3.4. Attack Graphs

AGs are directed graphs used to describe attack processes as sequences of steps from an initial condition to a target goal, or more generally from multiple initial states to multiple goals. In particular, each path in the graph from an initial state to a goal is a possible successful attack process.

Many variations of AGs have been used in the literature. SecuriDN works with AGs whose nodes are attack steps, countermeasures or analytics. In particular, the attack steps are techniques from the MITRE ATT&CK framework and the analytics are inspired from MITRE CAR (for more details, see Section 6.2).

A directed edge from technique A to technique B defines a precondition: in order to be able to carry out technique B, the attack step A must have been successful. A directed edge from a technique to an analytic implies that the analytic is triggered by the exploitation of the technique. A directed edge from a countermeasure to a technique indicates that the countermeasure hinders the attack's success. See Section 5.2.2 for further details.

3.5. (Dynamic) Bayesian Networks

BNs are the most adopted formalism in the area of predictive and causal inference. In recent years, they have been proven to be a valuable tool for addressing security problems [35,36], especially related to cyberattack explainability and evidence correlation. A BN can be defined as a pair $N = \langle\langle V, E\rangle, P\rangle$, where $\langle V, E\rangle$ are variable nodes and edges of a Directed Acyclic Graph (DAG), respectively, and P is a probability distribution over V. To each node of the network $V = \{X_1, \ldots, X_n\}$, a Random Variable (RV) is assigned so that a probabilistic relationship can be specified between them (if an edge $e \in E$ exists between the nodes X_i and X_j, it means that X_i directly influences X_j). In the specific case of discrete RVs, each local distribution can be specified in a tabular form (a column for each combination of states of the parent variables) called a Conditional Probability Table (CPT).

To represent more complex time-dependent interactions, DBNs [37,38] introduce an explicit temporal dimension so that the model is composed of multiple time slices. In our specific use case, we chose 2-time-slice Temporal Bayesian Networks (2TBNs), where the set of nodes $\{X_1, \ldots, X_n\}$ is replicated over two consecutive time slices, t and t'. With these premises, it is possible to define intra-slice edges (e.g., an edge from a node X_i^t to a node X_j^t, with $i \neq j$) or inter-slice edges (e.g., an edge from a node X_i^t to a node $X_j^{t'}$, where i can be equal to j). Notice that DBNs are a *discrete* time model and we need to use them to model a *continuous* time reality. We will discuss this in Section 3.5.1 and later at the beginning of Section 6.6.

Compared with the classic BNs, where only predictive and diagnostic inference can be performed on a single static snapshot of the model, with DBNs it is possible to execute more diverse inference tasks depending on the placement of unobserved variables (e.g., associated to specific attack steps) and observed variables (e.g., representing security alarms that could be raised in a vulnerable system) within the timeline.

The possible inference tasks are:

- *Filtering*: computing the probability of an outcome at time t (now) given the evidence available from time 0 up to the current time slice. From a security perspective, this strategy can be used to monitor the current state of a potential cyberattack based on the collected information.
- *Prediction*: computing the probability of a future outcome at time $t + h$ (where $h > 0$) given the evidence available up to the current time slice t. This inference task could be useful for predicting a future state of a cyberattack based on currently known information.
- *Smoothing*: computing the probability of a past outcome at time $t - l$ (where $l > 0$) given the available evidence up to time slice t. This kind of inference task could be useful for answering queries about the preconditions of a specific attack step.

Another intriguing aspect of DBNs is that both structure and CPT parameters can be learned from time-series data or security logs, such as the ones generated by online systems. This can be achieved by using Expectation Maximization and other algorithms, such as the one presented in [39], and allows semi-automatic model generation and parameterization without restarting the model design from scratch. In this paper, the DBN is derived from the AG modeling the combinations of attack techniques compromising the system (Section 5.2.4), so we do not need the methodologies mentioned above.

3.5.1. Inference Algorithms

An exact algorithm for DBN inference is 1.5JT [37], which converts the DBN into a *Junction Tree* (JT): the DBN variables are grouped into clusters, which become the JT nodes [40]. Then the inference is actually performed on the JT. The following aspects rule the computing effort for the execution of 1.5JT.

- Variable distribution in clusters: the higher the quantity of variables inside every cluster is, the higher is the computing effort. This holds in particular for the *interface variables*, i.e., all the nodes having inter-slice connections, because they influence the variables in the next time step.
- Query time: due to the discrete time assumption, the inference requires the model to be evaluated at every time step until reaching the query time.
- Time discretization step (Δ): the choice of Δ is crucial for the quality of the approximation, but at a cost.
  - A small value of Δ increases the number of time steps and the computing time as a consequence, but the continuous time model is better approximated.
  - A large value of Δ accelerates the inference process, but the consequence is a less accurate model.

The *Boyen–Koller* algorithm (BK) [41] can be considered the approximate version of 1.5JT. BK is still based on JT, but the clusters are defined by the user, with a consequent degree of result approximation depending on how the variables are distributed in the clusters. There are two special cases: (1) the interface variables are concentrated in one cluster, so BK is equivalent to 1.5JT (exact BK), with no approximation; and (2) each interface variable is inside a different cluster (fully factorized BK), leading to the maximum approximation.

Other approximate algorithms, such as *likelihood weighting* [42] and *particle filtering* [43], exploit stochastic simulation.

We exploit 1.5JT for the experiments presented in Section 6.6 because exact inference can be performed in a reasonable time on the model under study for the chosen value of Δ.

4. SecuriDN and the Online Platform

Within our framework, various components cooperate to achieve the goal of providing a comprehensive attack emulation and detection platform in order to allow the security analyst to perform security assessment tasks on a specific online architecture. In the following part of this section, we will briefly analyze the components of the platform.

At the core of our platform there is the capability of performing an automatic distributed deployment of the components specified in the architecture, thanks to the Docker Swarm mode [44]. This specific functionality allows the management of multiple Docker Engines running on different machines, with the benefit of being able to seamlessly reconfigure the entire architecture based on the requirements of the security analyst.

To execute the attack steps and verify the robustness of the online infrastructure, an emulated attacker has been developed. Its functionalities are mainly based on the Metasploit [45] tool, which, thanks to its ready-to-use library of modules and payloads, really facilitates the exploitation of a wide variety of real-world vulnerabilities. In addition to the already existing attacks, we have also implemented more power-specific steps. The emulated attacker is capable of autonomously performing the prescribed attack process up to its predefined goal (we will see in Section 5 how the goal is defined).

The attacker's activity is detected through various monitoring tools such as the Linux Audit [46] framework, together with the *auditd* module of Auditbeat [47]. These tools are able to perform integrity checks on critical files (e.g., modifications within the .ssh folder) or highlight suspicious system activities, such as the ones presented in Section 6.2. The data collected by the monitoring system are stored in the OpenSearch [48] database, where analytics and filtering functionalities are implemented so that an alarm is raised when specific events take place or certain thresholds are exceeded. Such alarms are conveyed through a Kafka [49] broker to feed the detection models. The example described in this paper shows the use of DBNs in modeling the conditional dependencies between certain alerts and their associated attack steps (see Section 6 for further details on this aspect).

5. SecuriDN

5.1. DrawNET Modeling System

The DrawNET Modeling System (DMS) [50] is a customizable framework for designing and solving models expressed in any graph-based formalism. Its open architecture allows us to manage new formalisms or existing ones through an XML-based language family; then, models expressed through such formalisms can be built by DrawNET, the model editor. It is interesting that additions do not require recompiling the DMS source code. Examples of applications of DMS in the past are formalisms such as Petri Nets, Bayesian Networks and Fault Trees, all with the corresponding solvers [51,52].

DMS is written in Java and is based on the *DNlib* library. In the following, we detail the main levels of DMS's architecture (see Figure 2).

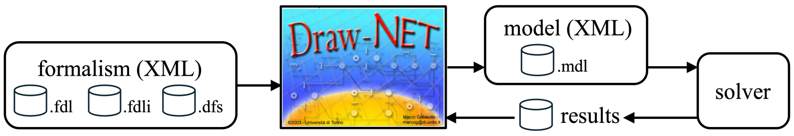

Figure 2. DMS general architecture.

5.1.1. Formalism Level

All the primitives to design a model are specified at the *formalism level*. To define a formalism, it is necessary to specify the tuple $F = \{E, P, C, S, H, T_P\}$, where E is the set of *elements*; P is the set of *properties*; C is the set of *constraints*; S is the *structure function* that associates elements to properties and inner elements; H is the *inheritance function* that specifies property inheritance for elements from specific (abstract) elements; and T_P is the *property typing function* that sets the type of each property.

Elements are the possible nodes and arcs in the model, and properties are their attributes. Elements also have graphical properties (shape, size, color, etc.). Properties are typed: integer, float, string, Boolean, etc. Logical propositions, constraints, describe consistency relations that must be satisfied by elements of a model. Finally, an element may contain inner elements in order to organize the model as a hierarchy of elements at different levels of depth.

Elements along with their properties (including the graphical ones) are saved to XML files.

5.1.2. Model Level

At the *model level*, the primitives defined at the formalism level are used to specify a model as a tuple $M = \{F, I, m_0, T, V, L\}$. Here, F is the formalism; I is the set of element instances, with every $i \in I$ representing an instance of an element of F; $m_0 \in I$ is the *main* element containing the elements of the main model, which in turn contain the elements of the submodels, in a recursive way according to the element hierarchy established in F; T is the *element typing function* that associates $i \in I$ with a (non-abstract) formalism element to which i corresponds; and the *assignment function*, V, specifies the property values, i.e., $V(i, p)$ is the value of property p of instance $i \in I$.

On DrawNET's GUI, a user selects one of the available formalisms, and the system loads its definition from the XML files, so the user can design models conforming to that formalism (Figure 2). The model is saved to an XML file.

5.1.3. Solver Level

At the *solver level*, any elaboration of the model can be carried out: conversion, analysis, simulation, etc. The user defines the results to compute on any model. When the solver is executed, the results can be shown by DrawNET at the end of the model solution (Figure 2).

5.2. The SecuriDN Tool

We implemented a prototype of the tool called SecuriDN, which allows us to define the architecture of an IT/OT system, the attacks to which the system is possibly vulnerable, and the parameters that characterize the various attack steps. Given these definitions, the tool generates models that allow us to analyze the behavior of the system. The implementation of SecuriDN is based on the DMS that provides the GUI and the library DNlib for model construction and manipulation. In particular, SecuriDN is based on the following formalisms and corresponding models.

5.2.1. Architecture Graph

The *Architecture Graph* (ArchG) consists of a set of assets (nodes) and associations (edges) between assets. As a first step, the user defines the ArchG. The assets currently foreseen in the formalism represent the main possible targets of an attack: computers, networks, applications, channels, etc. The assets are connected by arcs whose graphic style indicates the associations (connect, execute, communicate, attack, etc.), which represent potential means of attack propagation across assets. The constraints in the formalism rule the types of asset involved in a specific association. For example, a *connect* association can be set between a *network* asset and a *computer* asset. The other associations are presented in Section 6.5.1.

In addition to assets, the architecture contains two special nodes: attacker and goal. The attacker node is connected (through an *attack* association) to the asset where the attack begins, specifying the initial technique in a property of the node. In a similar way, the goal node is connected to the asset that is the final target of the attack, and has a property to indicate the technique to compromise the asset and the system as a consequence.

5.2.2. Local Attack Graph

The *local attack graph* (lAG) is a DAG that shows the attack processes that could be carried out against the asset, built as chains of MITRE ATT&CK techniques. For each asset, a lAG is defined by the user, modeling these combinations. The lAG can contain these types of nodes:

- An **internal technique** (represented by a simple circle) is a technique that takes place within the asset modeled by the lAG.
- An **external technique** (double circle) is a technique affecting another asset, but is enabled by one or more techniques taking place in the local asset. The external technique has a property specifying the path in the ArchG, going from the local asset to the assets where the external technique takes place. The path is expressed as a sequence of association types corresponding to the edges to be traversed in order to reach such assets.
- A **logical operator** (AND, OR) expresses the combination of two or more techniques.
- A **defense** (graphically represented by shield) represents a countermeasure, such as a firewall or an antivirus, able to mitigate or even inhibit an internal technique to which it is connected.
- An **analytic** (graphically represented by a notepad) represents an event that may be a clue about the exploitation of one or more techniques.

According to the constraints in the formalism, an oriented arc can go:

- From an internal technique to another technique (internal or external) to indicate that the first technique enables the second one;
- From an internal technique to a logical operator to indicate that the technique is combined with other ones;
- From a logical operator to a technique to indicate that a technique (internal or external) is enabled by a combination of internal techniques;
- From a defense to an internal technique to indicate that the defense mitigates the technique;
- From an internal technique to an analytic to indicate that the execution of the technique determines the production of the analytic.

5.2.3. Global Attack Graph

The *global attack graph* (gAG) is automatically obtained by running a solver that combines the lAGs of the individual assets present in the ArchG.

The generation process is accomplished through the following steps:

- **Union of lAGs:** a raw gAG is initially created as the union of the lAGs of all the assets in the architecture.
  The cost of this operation is linear in the total number of nodes (N_l) and edges (M_l) of *all lAGs*, since it just requires copies: $O(N_l + M_l)$.
- **Connection of lAGs:** if an external technique corresponds to an internal technique, and the corresponding assets are connected in the ArchG following the path associated with the external technique, then the external technique and the internal technique

are merged in one node in the gAG. If these conditions are not satisfied, then the external technique is removed from the gAG. This is done for every external technique in the gAG.

Each external technique, t_e, must be compared with each internal technique of an asset reachable from it. To determine reachability, a visit is carried out that works at a hybrid level, since it moves between the single components of the raw AGs (corresponding to the original lAGs), traversing edges from the ArchG. Then, the worst case complexity is $O((N_l + M_l + M_a)N_l)$, where M_a is the number of edges of ArchG, and N_l and M_l are the total number of nodes and edges of the lAGs: $O(N_l + M_l + M_a)$ the cost of the visit, since it amounts to visiting a graph with N_l nodes and $M_l + M_a$ edges. The factor N_l bounds above the number of external techniques, t_e.

Current annotations of nodes in the lAGs (properties) enable faster performance in the typical architectural topology in the context of power system networks by constraining the search, exploiting limits imposed by network specificities. The annotations that are now manually added by the cybersecurity analyst will be derived automatically by the tool in a forthcoming version. Notice in any case that the overall asymptotic complexity is not affected by these heuristics.

- **Identification of attacker and goal:** in the ArchG, the attacker node is connected to the asset where the attack begins, while the goal node is connected to the asset that is the final target. The attacker node and the goal node have a property to specify the initial technique and the final technique, respectively. The two nodes corresponding to these two techniques are identified in the gAG.
  The number of nodes of the raw gAG is $\Theta(N_l)$, with N_l the total number of nodes of all the lAGs. Identifying attacker and goal nodes has a cost linear in N_l: $O(N_l)$.
- **Reduction:** by visiting the gAG, all the paths from the initial technique to the final technique are identified. All the nodes and the arcs belonging to such paths are maintained, while all the other ones are eliminated from the gAG, thus obtaining the final, simplified gAG.
  The total number of edges of the raw gAG is $\Theta(M_l + M_a)$. The reduction requires a complete visit of the raw gAG from the attacker's node (the initial technique): the cost is $O(N_l + M_l + M_a)$.

5.2.4. Dynamic Bayesian Network

The gAG is then converted into a DBN with a compact representation where all associated state variables are binary. Each node in the gAG (technique, logic operator, defense and analytics) is translated to a DBN node, and each arc to a DBN arc.

In this way, not only is the dynamics of the whole attack described, as in the gAG, but also the underlying stochastic attack process is modeled. Technique nodes are enriched with a self-loop temporal arc to model the dependence of their state from the state at the previous time instant.

The production of the DBN requires creating $N_r + M_r$ DBN elements, where N_r is the number of nodes and M_r is the number of edges in the (reduced) gAG, with a self-loop for each internal technique, at most N_r self-loops. Notice that $N_r = O(N_l)$ and $M_r = O(M_l + M_a)$, so the total cost can be expressed as $O(N_l + M_l + M_a)$ in terms of the ArchG and lAGs.

A successfully executed technique influences the activation of the connected analytic to model the occurrence of an alarm. Through the CPT parameters, we configure the rates of false positives and negatives of each analytic. On the other hand, a defense node connected to a technique node influences the activation of the latter, reducing its probability of success possibly to zero; this models the mitigation or inhibition effect of the defense measure.

CPT parameters can be manually or automatically set. In the former case, using the GUI, the user can inspect each node of the gAG and compile the corresponding CPT. In the latter, various alternatives are possible. One possibility is that the user specifies an estimated mean Time to Compromise for each technique node of the gAG in the GUI, and from all these values the conditional probabilities of each node are automatically computed (see Section 6.6). Another option is that such parameters are learned by measurements from a real system or experimental testbed, or also extracted from synthetic simulation traces; in this case, no further input is required from the SecuriDN user because the learning process will be performed by external tools.

The DBN model is then used as a detection module of the monitoring and detection platform. The module can return predictive or diagnostic results conditioned by the evidence (observations) about the events occurring in the monitored system, collected through the platform. Several examples of the types of results provided by the analysis of the DBN can be found in [6].

6. Case Study

6.1. Case Study Architecture

Figure 3 shows the architecture we refer to. On the left, the figure depicts the network's hardware, and on the right we detail the relevant applications running on each host.

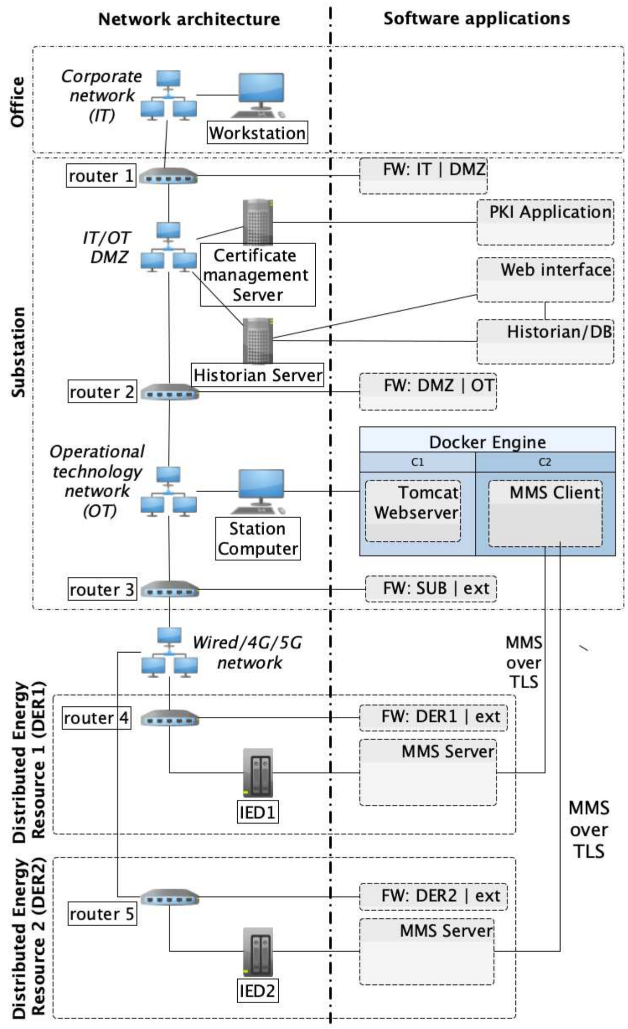

Figure 3. Case study architecture.

The networks, from the top, are the corporate network (IT), the DMZ and OT networks at substation level, and at the bottom the remote Distributed Energy Resources. The DERs are connected to the substation through an external network.

A DMZ separates the IT network from the OT one. The networks are pairwise connected by routers that filter traffic.

On the corporate network, we highlight a workstation that we define as the attack's starting point (see Section 6.5.1). On the DMZ, we assume, among others, two different hosts.

On the Historian server, a historian database is running; the database is updated from the OT network but it can be consulted from the IT network through a web interface running on the same host. The web interface requires no authentication, as it is meant only to view and not to modify data. Unfortunately, the web interface is poorly designed as no data sanitization is enforced (see Section 6.3).

The second host on the DMZ, the certificate management server, runs a PKI application. The service connects to the target hosts through HTTPS connections. To enable this, the firewall running on Router2 allows HTTPS connections to the OT network.

In the OT network, we focus on a station computer that runs two containerized services: a Tomcat web server implementing the local HMI for monitoring and control of the power area of responsibility, and an MMS client that connects to field devices to request measures and to issue commands.

In our case study, we assume two DERs in each of which a IED runs an MMS server which connects to the client on the station computer with MMS over TLS.

6.2. Monitoring System

In this network, a monitoring system has been implemented. On each server in the DMZ and OT networks, the following operations are monitored and reported:

- Shell execution (on a server, a shell should be executed only for maintenance, and therefore any such event must be reported).
- Remote shell session (a remote shell session should be reported and the source IP of the connection should be verified).
- Frequent failed login attempts (whenever a service allows remote access, a series of failed login attempts must be reported).
- Execution of suspicious commands (commands such as file system mounts, creation of new users and installation of new software, e.g., with apt-get, must be reported as they are not everyday operations).
- Access to files containing credentials (access by a non-intended actor should be considered suspicious; for instance if a server's private key is accessed by any other user).
- Integrity checks of critical directories (for instance, a directory containing the certificates of trusted CAs or the `.ssh` directories of privileged users, or of any user on critical hosts).

Moreover, we assume that specific application level analytics are implemented on the MMS traffic. In particular, the following are monitored:

- The coherence of the measures received from the field devices;
- The coherence of the commands issued by the SCADA system.

If the sequence of measures/commands exhibits significant deviations from the expected behavior, an alarm is raised.

Most of the monitoring we implemented is inspired by the MITRE CAR framework [34] (see Section 3.3), with the significant difference that the analytics from CAR are mostly implemented for Windows systems, whereas we work in a Linux environment. For instance, monitoring of shell execution is inspired by analytic CAR-2014-04-003, monitoring of remote shell session by CAR-2014-11-004, and so on. As we mentioned in Section 4, the analytics we use are implemented using the Linux Audit framework, together with the *auditd* module of Auditbeat.

The application level analytics constitute an exception: we designed them specifically for the power system environment.

6.3. Attacks Description

We consider that the attackers can use the following attack steps. We refer for each attack to the ATT&CK matrix technique it implements and the tactic it realizes [32,33]. Notice that a single technique can be used for various tactics; in case of ambiguities, we choose according to the objectives we intend our attackers to achieve.

Scanning IP blocks (Enterprise ATT&CK matrix—Tactic: Reconnaissance—Technique: Active Scanning). The attackers gather information on IP addresses actually used by hosts to start their attack.

Vulnerability scanning (Enterprise ATT&CK matrix—Tactic: Reconnaissance—Technique: Active Scanning). The attackers test potential victims for vulnerabilities.

Private keys (Enterprise ATT&CK matrix—Tactic: Credential Access—Technique: Unsecured Credentials). Attackers obtain a private key that is insecurely stored and the corresponding certificate.

Installing a rogue CA (Enterprise ATT&CK matrix—Tactic: Credential Access—Technique: Modify Authentication Process). Attackers install their own rogue CA among trusted CAs of the victim.

Password guessing (Container ATT&CK matrix—Tactic: Credential Access—Technique: Brute Force). Preconditions for this attack are a remote service with weak administrator credentials, in particular running in a container. The attackers try a set of common passwords and obtain access as an administrator on the victim host.

SQL injection (ICS ATT&CK matrix—Tactic: Lateral Movement—Technique: Exploitation of Remote Services). Precondition for this attack is that on the victim machine a vulnerable web application is running, which accesses an SQL database based on user input. The attack step exploits the fact that the web application accepts user input without sanitizing it. The attack has a high impact if the database runs as a SYSTEM user, with full privileges. In our scenario, attackers use an SQL injection to run system commands as the SQL user.

Unix shell (ICS ATT&CK matrix—Tactic: Execution—Technique: Command Line Interface). Precondition for this attack is that the attackers have user access on the target host. They obtain a shell on the host.

Remote SSH connection (Enterprise ATT&CK matrix—Tactic: Command and Control—Technique: Remote Access Software) Precondition for this attack is that the attackers have obtained a user's SSH private key on the remote target host or installed their own public key. They open an SSH connection to the host.

Docker escape (Container ATT&CK matrix—Tactic: Privilege Escalation—Technique: Escape to Host). This attack exploits the CVE-2019-14271 [53,54] vulnerability of Docker v.19.03.0. Preconditions are that the attackers have control of a container with root privileges. In this case, they can modify a system library file in the container to a malicious one, and, when an unaware user copies a file from the machine to the compromised container, the attackers gain access to the whole host with root privileges.

SSH Authorized Keys (Enterprise ATT&CK matrix—Tactic: Persistence—Technique: Account Manipulation). An attacker that has obtained a shell as user U on a host that runs SSH installs its own SSH public key to obtain easy future access to the host with the same privileges already obtained.

Adversary in the middle, on a TLS secured channel (ICS ATT&CK matrix—Tactic: Collection—Technique: Adversary-in-the-Middle). Preconditions are that the attackers have full control of the victim host, have hold of the client's (resp. server's) private key and have installed their rogue CA on the client (resp. on the server). Alternatively, they may have hold of both the client's and server's private keys or have installed their rogue CA on both the client and the server. On the attackers' host, malicious software handles the traffic. The attackers first take advantage of the iptables service running on the host to hijack traffic to/from specific IPs or ports. Then they send a RESET to terminate a possibly running connection, and then, when the client starts a new connection, mount an adversary in the middle attack, using SSLsplit [55]. When the client contacts the server to establish a new communication, the attackers' malware will, e.g., impersonate the server at the client using a fake certificate signed by the attackers' CA, and it will impersonate the client at the server using the stolen client private key and certificate. (In case the preconditions are different, according to the various options listed above, impersonation will work differently).

Reporting message/command injection (ICS ATT&CK matrix—Tactic: Impair Process Control—Technique: Spoof Reporting Message/Unauthorized Command Message). We specifically consider injecting packets in an MMS over TLS channel. Preconditions are that the attackers can mount an adversary in the middle attack over TLS. Then, the attackers' malicious software that handles the hijacked traffic is a package that either injects unauthorized commands for the server or spoofs measure-reporting messages from the server to the client. The consequence sought by the attackers is a DER failure. In the case of a command injection, the server will relate the command to the IED, which will operate according to the attackers' instructions, causing a failure of the DER. In the case of a reporting message injection, the aim is to cause the client to have an altered view of the system's state and to react in a way that is appropriate for the view but not to the real system state, thus again causing the DER's failure.

6.4. Monitoring System and Attacks

We discuss here which of the listed attack steps can be exposed by the implemented monitoring system. Table 1 summarizes the picture. Most entries in the table need no comment but some do. In particular,

- The *escapeHost* implementation we consider (Docker escape) requires that the attackers modify a specific system library file; this operation would be detected monitoring the integrity of critical directories;
- The same Docker escape implementation requires a mount operation, and therefore the analytic "execution of suspicious commands" would raise an alert in case this attack step were attempted by attackers;
- Application level coherence monitoring on reporting messages and commands could expose measure and command injection, unless the attackers are very careful to make slow changes that raise no suspicions;
- An integrity check on the directory in which the iptables data are stored would expose any changes to iptables' configuration;
- We assume for the sake of the case study that no monitoring to detect suspicious traffic related to scanning is implemented.

Table 1. Attacks and analytics.

Attack Steps	(Abbreviations)	Analytics	(Abbreviations)
Scanning IP blocks	(*scanIP*)		
Vulnerability scanning	(*scanVuln*)		
Private keys	*unsecCred*	access to files containing credentials	(FILEACCESS)
Installing a rogue CA	(*modAuthProc*)	integrity checks of critical directories	(INTEGRITY)
Password guessing	(*bruteForce*)	frequent failed login attempts	(LOGINFREQUENCY)
SQL injection	(*rmtSrvc*)		
Unix shell	(*shell*)	shell execution	(SHELLEXECUTION)
Remote SSH connection	(*rmtSrvc*)	remote shell session	(RMTSHELLSESSION)
Docker escape	(*escapeHost*)	execution of suspicious commands	(SUSPICIOUSCMD)

Table 1. *Cont.*

Attack Steps	(Abbreviations)	Analytics	(Abbreviations)
SSH Authorized Keys	(*SSHkey*)	integrity checks of critical directories	(INTEGRITY)
Adversary in the middle, on a TLS-secured channel	(*AITM*)		
Reporting message injection	(*spoofRepMsg*)	coherence of measures	(MEASCOHER)
Command injection (*unauthCmdMsg*)	coherence of commands	(CMDCOHER)	

6.5. Modeled Scenario

We have built in SecuriDN the architecture of Figure 3. The lAGs of the various assets describe the attacks we take into account for our case study (Section 6.3), together with the analytics from our monitoring system (Section 6.2) and their relationships (Section 6.4).

6.5.1. Architecture Graph

Figure 4 shows the GUI of SecuriDN, where we can notice the drawing area and the panels to select nodes and edges.

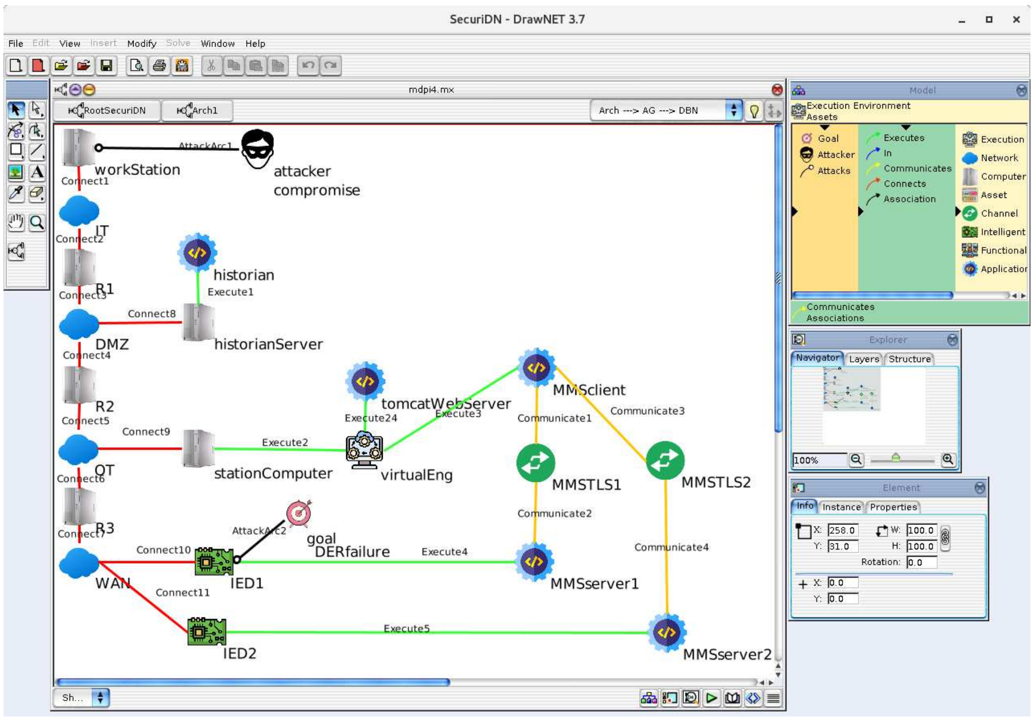

Figure 4. SecuriDN's GUI, showing the ArchG graph of our case study. Notice the drawing area to the left and the panels to select nodes and edges to the right.

In particular, in Figure 4 we can see the ArchG of the case study. Different icons are used to draw different types of assets, to make the graph more intuitive. In particular, we have

- Several computers (workstation, routers R1, R2, R3, historian server, station computer);
- Several networks (IT, DMZ, OT, WAN);
- One execution environment (virtual engine);
- Several applications (historian, Tomcat web server, MMS client, MMS server 1, MMS server 2);
- Two channels (MMS TLS 1, MMS TLS 2);

- Two IEDs (IED 1, IED 2).

In the ArchG formalism, other types of asset are available, as shown in the panel on the right, in Figure 4. In addition to the nodes corresponding to assets, we have the nodes that specify the adversarial setting the security analyst wants to analyze:

- The attacker node specifies, for the analysis, the asset that is assumed to be initially compromised (*compromise*)—in our case study, it is a corporate work station, but it could be external to the network;
- The goal node is the asset the security analyst wants to focus on; possible compromise of this asset will be central in the analysis carried out by the model—in our case study the goal is set as one of the DERs, *DERfailure* in IED 1.

The edges have different colors and labels in order to identify the possible associations between assets:

- *Connect* (red) is the association between a computer and a network. For example, between workstation and R1.
- *Execute* (green) is the association between a computer and an application, between a computer and an execution environment, or between an execution environment and an application. Examples are the associations between the historian and historian server, between the station computer and virtual engine, and between the virtual engine and Tomcat web server.
- *Communicate* (yellow) is the association between an application and a logic channel. For example, MMS TLS 1 is associated with the MMS client and MMS server 1.

6.5.2. Local Attack Graphs

After the design of the ArchG, the combination of techniques inside each asset is modeled by the corresponding lAG, as described in Section 5.2. As an example, let us consider the lAG of the historian server, depicted in Figure 5. Table 1 lists the abbreviations used in SecuriDN for techniques and analytics.

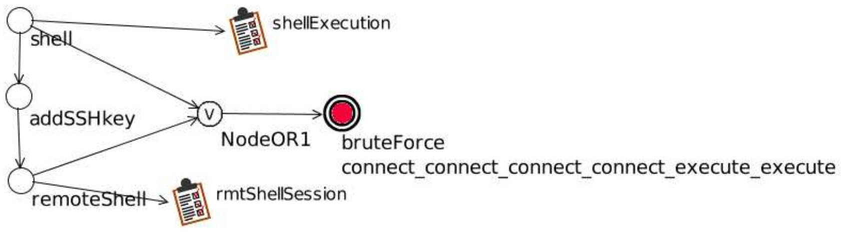

Figure 5. The lAG of the historian server (the red node will be merged with the corresponding red node in Figure 6 during the construction of the gAG). If the attackers have the ability to run a *shell* on the machine, they can install an SSH key. With an SSH key installed, they can connect also remotely (*remoteShell*). Notice the two analytics SHELLEXECUTION and RMTSHELLSESSION. From this asset, the attackers can try *bruteForce* on a remote device reachable via a network connection.

We can notice the internal techniques (circles), the logical operators (AND/OR), the external techniques (double circles) and the analytics (notepads). In particular, *shell* enables *SSHkey*, which in turn enables *remoteShell*, and they are all internal techniques. The analytics SHELLEXECUTION and RMTSHELLSESSION are generated by the occurrence of *shell* and *remoteShell*, respectively. Through the OR logical operator, *shell* or *remoteShell* enables the external technique *bruteForce*, taking place in the assets reachable by following a path composed of four *connect* associations and two *execute* associations. In the ArchG (Figure 4), such a path exists from the historian server to the Tomcat web server. In the lAG of the latter asset, we can see the corresponding internal technique, *bruteForce* (Figure 6).

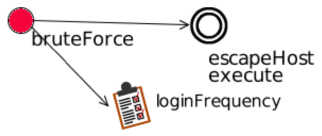

Figure 6. The lAG of the Tomcat web server shows that initial access can be obtained through *bruteForce*, after which an *escapeHost* can be carried out on the virtual environment on which the server is running. The analytic LOGINFREQUENCY exposes a *bruteForce* attempt. The red node will be merged with the corresponding red node in Figure 5 during the construction of the gAG.

Another example is the lAG of the MMS client, depicted in Figure 7. The internal techniques *unsecCred* and *modAuthProc* generate the analytics FILEACCESS and INTEGRITY, respectively. Through the AND logical operator, the same techniques enable *credAcc*, which in turns enables the external technique *AITM* taking place in the asset MMS TLS 1 (Figure 8), reachable from the MMS client through one *communicate* association (Figure 4). The technique *spoofRepMsg* generates MEASCOHER and enables *DERfailure* taking place in the asset IED 1, reachable through a path composed of two *communicate* associations and one *execute* association (Figure 4).

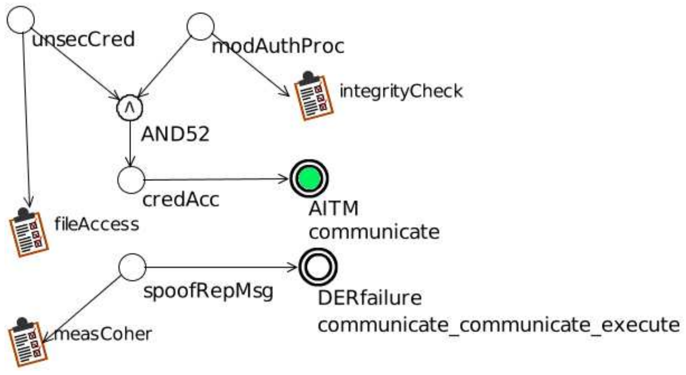

Figure 7. The lAG of the MMS client is more articulated (the green node will be merged with the corresponding green node in Figure 8 during the construction of the gAG). If attackers can steal credentials and manipulate trust (*unsecCred* and *modAuthProc*), they can then carry out an *AITM* on the communication channel. Or, if they manage to spoof reporting messages, they can induce the MMS client to cause a *DERfailure*.

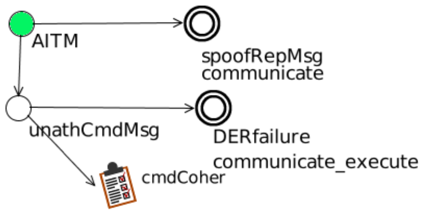

Figure 8. The lAG of the MMS over the TLS channel describes the fact that, in case an *AITM* is carried out on the channel, attackers can then send crafted reporting messages to the client (*spoofRepMsg*) or send unauthorized command messages (*unauthCmdMsg*) directly to the server to cause a *DERfailure*. The green node will be merged with the corresponding green node in Figure 7 during the construction of the gAG.

6.5.3. Global Attack Graph

The raw gAG is obtained (see Section 5.2) as the union of all lAGs. Then, in Figure 9, we can see the gAG obtained by merging the external techniques with the corresponding internal techniques. For example, the external technique *historianServer_bruteForce* (red node in Figure 5) is merged with the internal technique *tomcatWebServer_bruteForce* (red node in Figures 6 and 9). In the same way, the external technique *MMSclient_AITM* (green node in Figure 7) is merged with the internal technique *MMSTLS1_AITM* (green node in

Figures 8 and 9). Still, in Figure 9, the nodes and the edges not leading to the final technique (*DERfailure*) from the initial technique (*compromise*) have been removed, as explained in Section 5.2 (the nodes *compromise* and *DERfailure* are yellow).

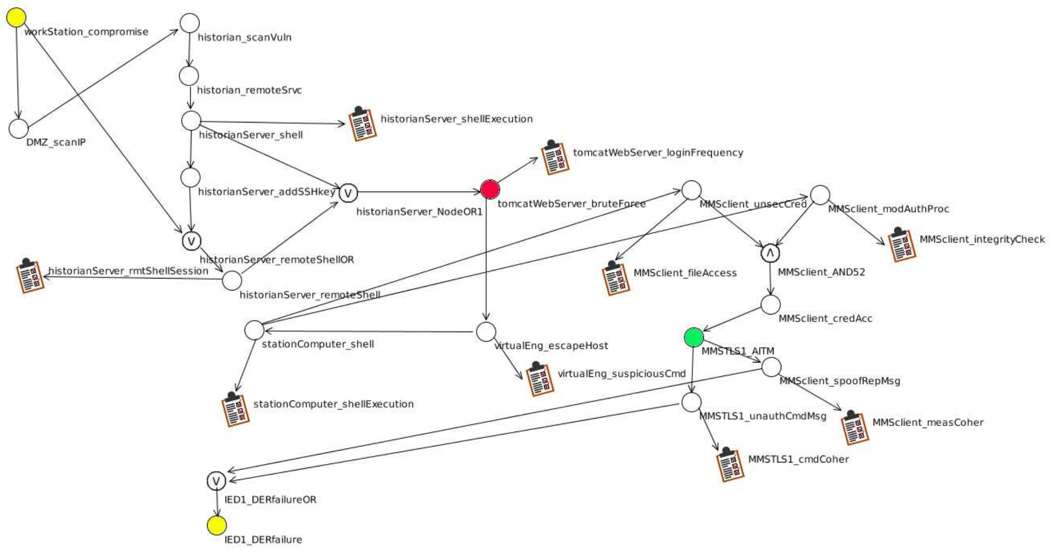

Figure 9. The final gAG obtained merging lAGs and pruning the nodes not leading to the goal node. The red and green nodes are examples of merged nodes (see Figures 5–8). The yellow nodes indicate the initial technique and the final technique. Note that the figure shows precisely the output of SecuriDN (the graphical aspects will be improved in a later version of the prototype).

6.5.4. Dynamic Bayesian Network

The DBN in Figure 10 is derived from the final gAG in Figure 9. Both models share the same graph topology, but in the DBN the temporal arcs (blue) are added for the techniques because of their probabilistic evolution during time (Section 5.2).

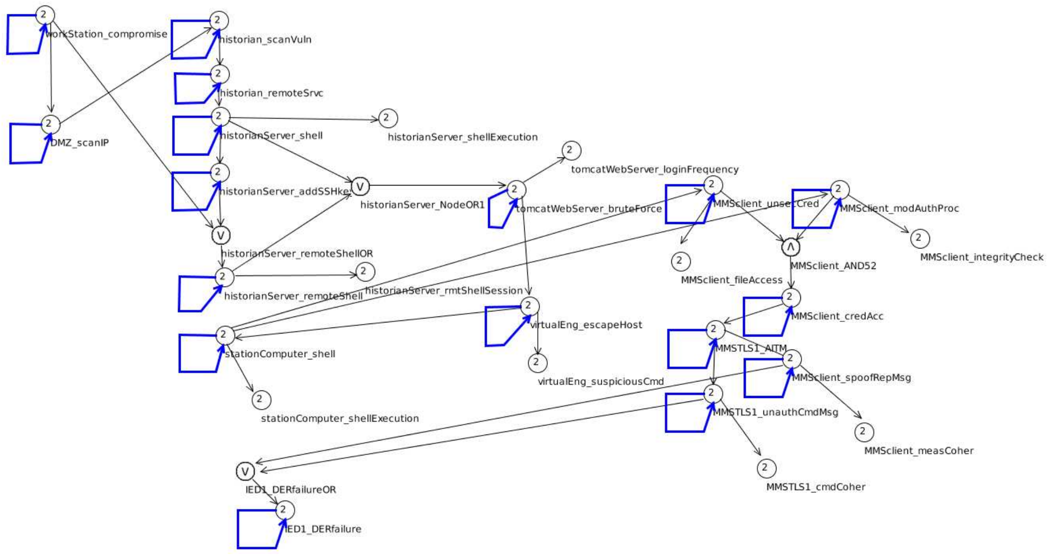

Figure 10. The DBN derived from the final gAG. The nodes containing the symbol "2" are binary random variables representing techniques, analytics, or defenses, all characterized by two possible states (not occurred, occurred). The logic nodes (AND, OR) contain the corresponding Boolean operator ($\wedge$, $\vee$). The oriented black arcs indicate that a node influences another node. Finally the blue loops represent the temporal arcs to model probabilistic evolution over time.

6.6. Experiments and Results

In our experiments, we have parametrized the DBN on the basis of estimated mean completion times of the attack steps. Notice that the DBN is a *discrete* time model, whereas we need to model the *continuous* time reality. To approximate continuous time, we assume that each time step of the DBN has a constant duration and re-scale all probabilities with respect to this value. In this way, the elapsed times between two consecutive state transitions in the continuous model are approximated by a geometric distribution in the DBN.

Based on evidence from the monitoring system, the derived DBN can then be used to forecast the most likely adversarial activity, along with a probability distribution of expected upcoming attack steps. Knowledge of the time available before the attack actually affects core functionalities enables cybersecurity analysts to evaluate the most appropriate response.

The mean times to compromise (TTC) we adopted are reported in Table 2. They are expressed in a 10 min unit.

For *scanIP*, *scanVuln* and *bruteForce*, we assumed that the attackers slow their activity in an attempt to avoid detection (but not enough for *bruteForce* to escape our monitors).

The *escapeHost* technique is very slow because the attackers must wait until an unaware user copies a file for their exploit to succeed. The mean TTC for this step should realistically be even longer, but we limited it to bound the total time of the attack process for better readability of the diagrams plotting the results; *unauthCmdMsg* takes a long time too.

Table 2. Mean times to compromise (TTC), expressed in 10 min units—see Table 1 for the meaning of the technique names.

Technique	Mean TTC	Technique	Mean TTC	Technique	Mean TTC
scanIP	3	*rmtSrvc*	2	*SSHkey*	1
scanVuln	3	*shell*	0.5	*AITM*	4
unsecCred	1	*remoteShell*	0.5	*spoofRepMsg*	15
modAuthProc	1	*escapeHost*	50	*unauthCmdMsg*	40
bruteForce	6				

The implemented analytics are almost perfect (the assumed error probability is 0.01).

We have run our automated attacker in the testbed with different strategies and with different configurations of the monitoring system.

Notation: in the following, we have abbreviated the names of the nodes of the DBN for the sake of the presentation. The abbreviated names of the techniques from Table 1 are completed, prepending *HS* for *historianServer*, *WS* for *tomcatWebServer*, *cli* for *MMSclient*, *VE* for *virtualEng* and *MMS* for *MMSTLS*. Other simplifications are self-explanatory.

6.6.1. Experiment 1: No Monitoring Implemented

We have a first reference experiment in a setting in which the monitoring system is not implemented, to evaluate how the DBN expresses the constraints among attack steps, as defined by preconditions.

The results are shown in Figure 11. We show here only the probabilities of some of the DBN's nodes to stress our points.

The order in which the curves start to grow reflects the constraints and the TTC: a remote shell session on the historian (*HS_remoteShell*) can be opened only after an initial reconnaissance (*DMZ_scanIP*) has taken place. The injections of false reporting messages (*cli_spoofRep*) and unauthorized commands (*MMS_unauthCmd*) are the final ones in the attack process, and, therefore, the probability curves for these two techniques start growing much later (notice the gap). Moreover, since *MMS_unauthCmd* has a much higher TTC than the other techniques considered in the figure, the rate of growth of its probability is the slowest. On the other hand, the probability of a failure of the DER

(*IED1_DERfailure*) is higher than the latter two since it can occur when either *cli_spoofRep* or *MMS_unauthCmd* occur.

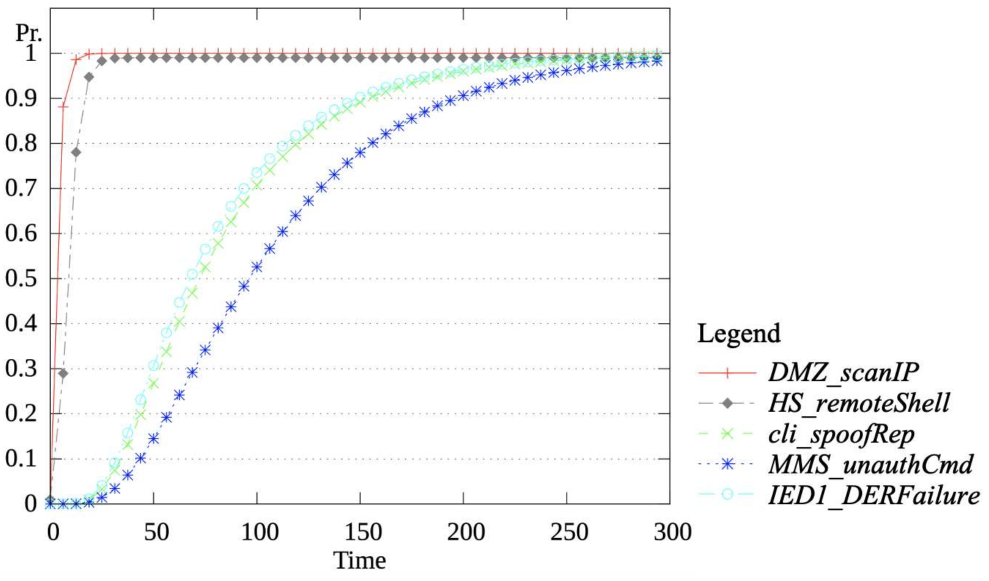

Figure 11. *Experiment 1.* Probabilities of successful technique exploitation with no evidence (all analytics disabled).

6.6.2. Experiment 2: Detection of Different Attack Processes

In our second experiment, monitoring is fully implemented. We had the automated attacker scan the network and carry out an SQL injection on the historian to open a shell as a privileged user. From that new foothold, the attacker carried out a password-guessing attack on the Tomcat web server and, having gained control of the container on which the server runs, it completed its attack with a Docker Escape.

Our monitoring system detected at time $t = 9$ a shell execution on the historian (analytic HS_SHELLEXEC), at $t = 16$ frequent failed attempts to connect to the web server as an administrator (analytic WS_LOGINFREQUENCY), and at $t = 56$ suspicious commands on the machine that hosts the container (analytic VE_SUSPCMD).

Figure 12 shows the probabilities of some DBN nodes.

We have not plotted the probability that a shell has been opened on the historian, but as a consequence of the first alert (HS_SHELLEXEC) we see that the probability that a reconnaissance activity has taken place shoots to 1, to warn the security analyst that hostile activity is taking place. Notice that we have not implemented a monitor to detect a scan of the network, yet the DBN informs us that it must have happened, because in our model we defined reconnaissance activity as a precondition of any attack. A more detailed model could take into account a variety of reconnaissance techniques.

Also notice that the probability of a scan was in any case increasing before $t = 9$, when the first alert was triggered, to reflect the fact that a security analyst cannot relax and sit back but must always take into account that attacks can happen at any time.

The second alert that goes off, WS_LOGINFREQUENCY, at $t = 16$, specifically detects that our automated attacker is trying a dictionary attack on the web server. Correspondingly, in Figure 12, the probability that this technique is being used (*WS_bruteForce* in the figure) suddenly jumps to 1.

Finally, the DBN interprets the evidence VE_SUSPCMD at $t = 56$ as a sure sign that the attacker has managed an *VE_escapeHost*.

Notice that the probability that a shell has been opened on the substation computer (*SC_shell*) remains 0 throughout our observation interval. This is because the event is monitored and the corresponding analytic does not raise any alert.

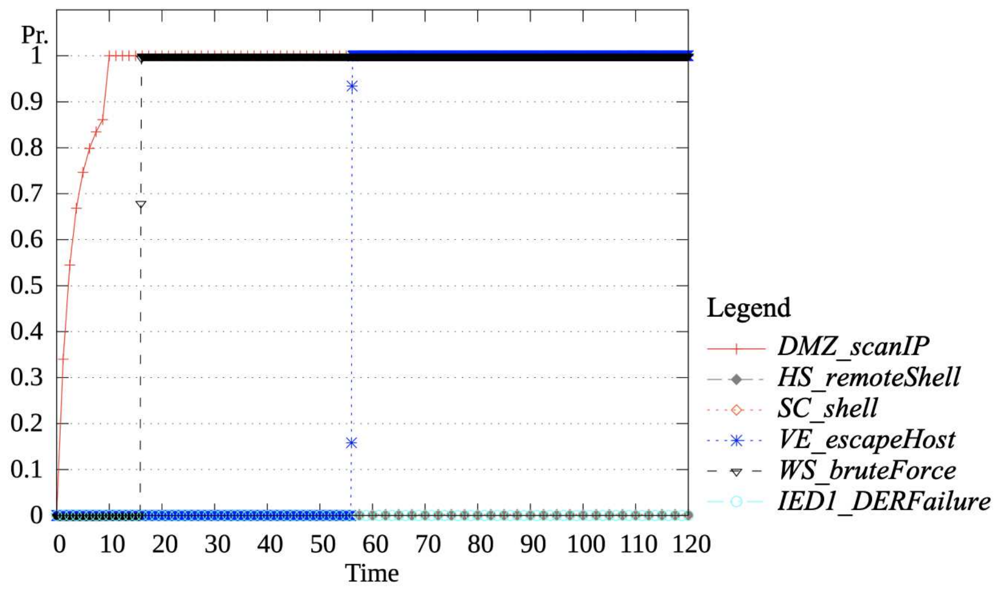

Figure 12. *Experiment 2–part 1.* Probabilities of successful technique exploitation with fully implemented monitoring system, and partial execution of the attack process. Alerts: HS_SHELLEXEC at $t = 9$, WS_LOGINFREQUENCY at $t = 10$, VE_SUSPCMD at $t = 56$. All other analytics raise no alerts.

In the second part of this experiment, we explore how our detection model reacts to a different attack path. We observe our automated attacker after it has already installed an SSH key on the historian server's host and starts its attack with a remote connection to that host. We disabled the monitor HS_SHELLEXEC to emulate the fact that we became aware of the adversarial activity only later.

The automated attacker then carries out the dictionary attack on the web server and does not proceed further in the observed time interval.

Figure 13 reports the probabilities of the same techniques as Figure 12 to compare the results.

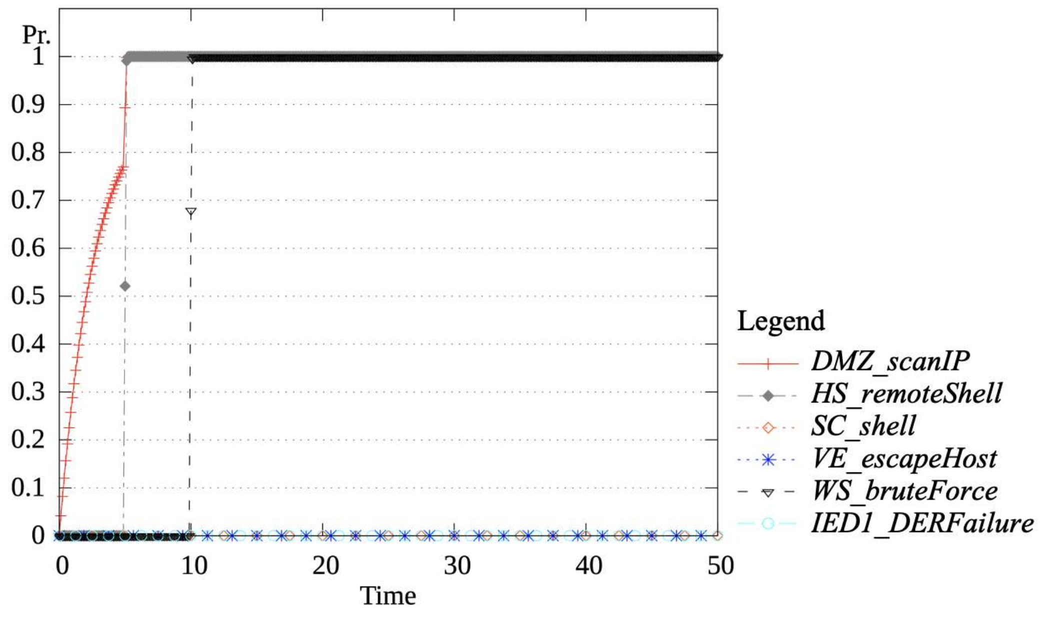

Figure 13. *Experiment 2–part 2.* Probabilities of successful technique exploitation, when the observed attack process starts with a remote connection to the historian server. Analytic HS_SHELLEXEC not active. Alerts: HS_RMTSHELL at $t = 5$, WS_LOGINFREQUENCY at $t = 10$. All other analytics are implemented but raise no alerts.

We have that a remote shell on the historian server is detected at time $t = 5$ (analytic HS_RMTSHELL). At that time, the probability of the monitored event (HS_RMTSHELL) correspondingly jumps to 1. Also, the probabilities of previous steps (we plotted in Figure 13 again *DMZ_scanIP*) go to 1, as we observed in the previous experiment.

Again, we see how the model correctly detects the dictionary attack on the web server at $t = 10$. No more adversarial activity takes place in the observed interval.

6.6.3. Experiment 3: Early Detection and Forecasting

In our final experiment, some analytics are disabled: HS_SHELLEXEC, SC_SHELLEXEC, CLI_FILEACCESS and MMS_CMDCOHER. As a consequence, we cannot detect the launch of a local shell, neither on the historian server nor on the station computer, and we are not monitoring file access to the MMS client's credentials, nor the coherence of commands to the IED at the DER.

The attacker's activity is again observed from the moment in which the attacker creates a remote shell on the historian server. In the first part of this experiment (Figure 14), we assume that the security analyst is observing the console of the detection system at time $t = 30$. Therefore, the available evidence is that up to that instant; after that, the evidence is undefined.

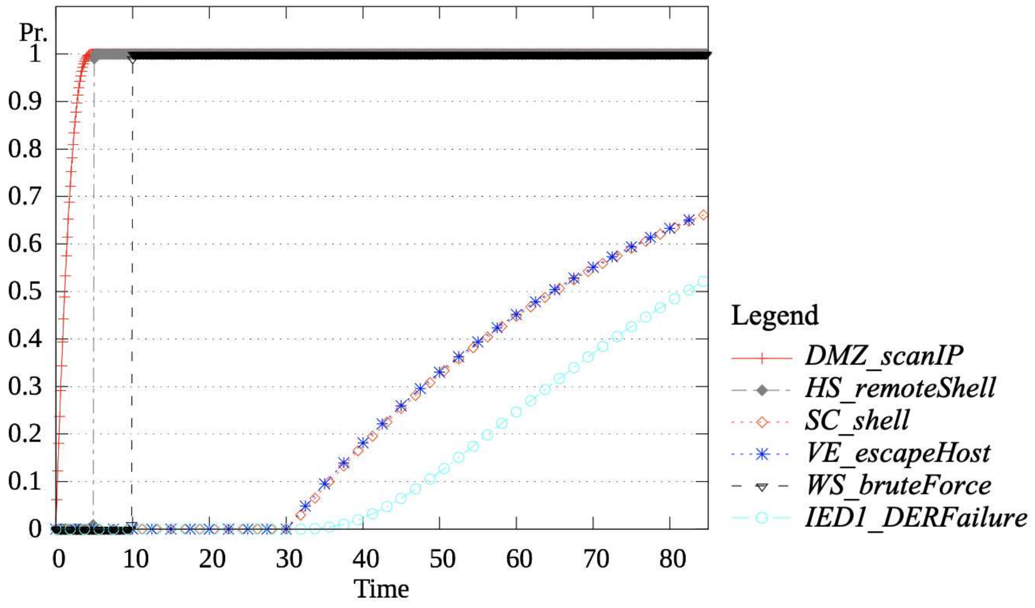

Figure 14. *Experiment 3 part 1.* Probabilities of successful technique exploitation with observations until time $t = 30$ and predictions after $t = 30$. Analytics HS_SHELLEXEC, SC_SHELLEXEC, CLI_FILEACCESS and MMS_CMDCOHER not active. Analytics Alerts: HS_RMTSHELL at $t = 5$, WS_LOGINFREQUENCY at $t = 10$. Analytics VE_SUSPCMD, CLI_INTEGRITY, CLI_MEASCOHER detect no suspicious activity. After $t = 30$, all analytics undefined.

The automated attacker behaves as in part 2 of Experiment 2.

The significant difference between the current experiment and the previous one is that in the previous we assume that the analyst is observing the console of the detection system at time $t = 50$ (the last instant plotted in Figure 13). This implies that, for the whole time interval considered there, evidence from all implemented analytics is known.

Correspondingly, in Figure 14, all probabilities up to $t = 30$ are exactly the same as in Figure 13: the detection system has analyzed the evidence and proposes to the security analyst its inference on the attacker's activity until that moment.

After $t = 30$, in the current experiment (Figure 14), in absence of new evidence and based on what has been seen so far, the detection model *predicts* the probabilities of subsequent adversarial activity. Since the web server has been compromised at $t = 30$, it is increasingly likely over time that the attacker will escalate privileges escaping to the host (*VE_escapeHost*), then open a shell on the station computer (*SC_shell*) and eventually undermine the DER's functionality (*IED1_DERfailure*).

In contrast, in Figure 13, all these further steps keep having a probability of 0 until the final time instant considered ($t = 50$), since the observation instant is $t = 50$ and it is known that no further alerts have been raised.

Figure 15 shows again a plot of a completely known time window: the observation instant is now $t = 85$. The attacker carries out an attack, reaching the final goal, choosing to spoof reporting messages once it has managed to intrude into the MMS over TLS communication (AITM).

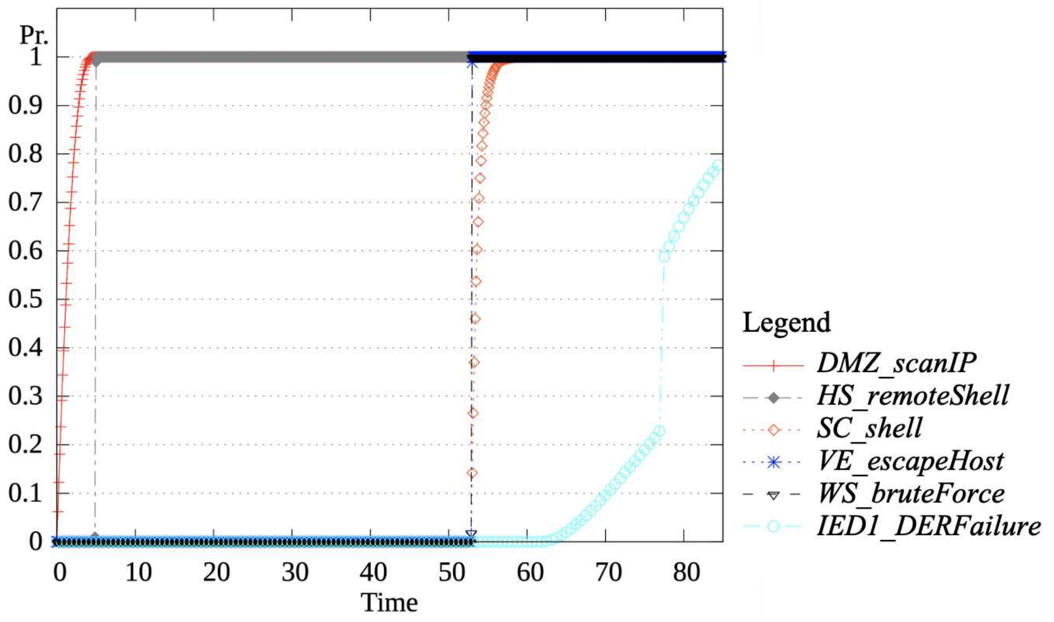

Figure 15. *Experiment 3–part 2.* Probabilities of successful technique exploitation with observations until time $t = 85$. Analytics HS_SHELLEXEC, SC_SHELLEXEC, CLI_FILEACCESS and MMS_CMDCOHER not active. Alerts: HS_RMTSHELL at $t = 5$, WS_LOGINFREQUENCY at $t = 10$, VE_SUSPCMD at $t = 55$, CLI_INTEGRITY at $t = 59$ and CLI_MEASCOHER at $t = 77$.

The monitoring system raises the same two alerts as in Figure 14, and then the alerts VE_SUSPCMD at $t = 55$ (detecting a suspicious command, given by the attackers to complete the escape to host), CLI_INTEGRITY at $t = 59$ (detecting that CA certificates trusted by the MMS client have been tampered with) and CLI_MEASCOHER at $t = 77$ (detecting an incoherence in the series of measures received by the MMS client from the server, due to an injection of reporting messages).

Comparing Figures 14 and 15, we notice how the detection tool gives an early warning.

As an aside, notice that shortly after $t = 60$, even before the final alert is raised at $t = 77$, the detection tool is warning that the DER functionality might be at risk. This is because we assumed that the analytic MMS_CMDCOHER (detecting a command injection attack) is not implemented, and therefore there is a possible final step of the attack process that is not monitored.

We conclude our analysis, pointing out that our initial experiment gives the output of the detection system at the very beginning of the observations: Figure 11 can be interpreted as the output of the detection tool at $t = 0$ when no evidence has been observed yet, and all probabilities represent a forecast of adversarial activity based on the structure of the DBN produced by SecuriDN and the expert insight on attack steps codified in the CPTs.

7. Conclusions and Future Works

The platform we have developed, and that was used for the case study presented in this paper, can support cybersecurity analysts in the early detection of adversarial activity and in the assessment of the cybersecurity posture of electro-energetic systems.

The realistic case study presented in this paper shows how the quantitative metrics provided by the detection model may give a useful insight into the evolution of the attack process according to the evidence collected by the monitoring system, and can be used to forecast the most likely adversarial activity and evaluate the system security level.

In this paper, we have described in detail one module of the platform: SecuriDN; its aim is to make it easier for cybersecurity analysts to define in a modular (bottom-up) way the model of the assets under study and of the possible attacks threatening them. SecuriDN is flexible because the library of elements to be included in the models can be expanded; moreover, it can apply several transformations on such models by integrating a number of so-called *solvers* that can produce the input for other modules in the platform. These features make it suitable for integration with other cybersecurity tools and frameworks. Since SecuriDN is completely customizable, it can be programmed to import and export architectures, and attack graphs and DBN models in the native XML format; moreover, implementing appropriate solvers, different formats can be chosen for export, and other artifacts of interest can be produced.

In the case study, SecuriDN has been used to produce a DBN for offline what-if analysis. The DBN model is conceived to be aligned with evidence collected by the online monitoring platform. In future experiments, the data from the running platform will be exploited to refine the DBN model and iterate the what-if analysis. In this paper, the practical application of SecuriDN was demonstrated in a simplified, but still realistic, case study of a multi-step attack process, starting from a compromised workstation in the corporate network and aiming to cause a manipulation of DER measures. The considered attack includes techniques taken from both Enterprise and ICS ATT&CK matrices realizing different stages of a cyber kill chain, such as reconnaissance, credential access, lateral movement and impairment of process control. From the proper composition of assets coming from a predefined and customisable library, SecuriDN allows us to automatically generate a gAG describing the whole attack process, and the corresponding DBN. Although SecuriDN is still a prototype, it currently supports the analysis of diverse cybersecurity scenarios.

In the future, we will carefully evaluate SecuriDN through the development of several case studies of increasing size and complexity, which will guide the tool development plan. Possible limitations of SecuriDN that we will explore are the user friendliness of the GUI (limited to its current usage within applied research projects); the maximum model size that can be built without overcoming the current DrawNET limits; the memory and computational resources required for the generation of the final attack graph and DBN; and the expressive power of the formalism currently employed to describe the scenarios (assets, configuration and corresponding attacks).

To improve usability, we are planning to create a library of attack graphs for each asset, which can be imported by the user to compose more complex global attack graphs, taking in consideration additional potential vulnerabilities.

As stated in [56], the computational complexity of generating AGs can become a significant issue, especially for automatically created and highly connected topologies. An approach pursued in [57] exploits the divide-and-conquer methodology by introducing distributed firewalls between network partitions with the idea of reducing the number of paths exploitable by the attacker. As further clarified in Sections 5.2 and 6.1, the task performed by the security analyst of manually drawing the ArchG, paired with the introduction of security rules in between the various sub-networks, implicitly reduces the number of available paths to the attacker in the final gAG. For this reason, in this paper we did not consider it necessary to study the performance of our generation algorithm, as the main objective is to present SecuriDN functionalities. Notice, moreover, that, whereas SecuriDN is an offline tool, the models it creates are meant to be used for online detection.

This opens a different issue, on the real-time performance of DBN solvers. This point is out of the scope of this paper, which focuses on SecuriDN, but for completeness we presented in Section 3.5.1 various inference algorithms that can be applied, depending on the specific DBN size and precision requirements to speed up the solution.

The limits on the model sizes that SecuriDN inherits from the tool DrawNET on which it is based will be overcome in the future thanks to the new version of DrawNET that is now under development [58].

Future work will focus on implementing in SecuriDN the automatic generation of an Octave [59] script to perform the inferences on the DBN. The detection module in the platform will execute the script when alerts are triggered by the attack to update the inference according to the new collected evidence. Other additional SecuriDN *solvers* are being designed to support the seamless integration of the SecuriDN module with other modules of the platform, in particular with the attacks emulator (to be deployed in the RSE testbed) and with the simulator.

Author Contributions: Conceptualization, D.C., D.C.R. and L.E.; Methodology, D.C., D.C.R., L.E., G.F. and L.P.; Software, D.C.R.; Validation, G.D. and R.T.; Visualization: D.C. and D.C.R.; Investigation, D.C.; Writing—original draft, D.C., D.C.R., G.D., L.E., G.F., L.P., D.S. and R.T.; Writing—review & editing, D.C., D.C.R., G.D., L.E., G.F., D.S. and R.T; Project administration: D.C. and R.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work is original and has been supported by a joint collaboration between RSE S.p.A. and Università del Piemonte Orientale, financed by the Research Fund for the Italian Electrical System under the Three-Year Research Plan 2022–2024 (DM MITE n. 337, 15 September 2022), in compliance with the Decree of 16 April 2018.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: The authors would like to thank Alberto Livio Beccaria for implementing the *DNlib* library, and Marco Gribaudo for the development of *DrawNET*.

Conflicts of Interest: Authors Giovanna Dondossola and Roberta Terruggia are employed by the company Ricerca sul Sistema Energetico (RSE S.p.A.). The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest. The authors declare that this study received funding by the Research Fund for the Italian Electrical System under the Ministerial Three-Year Research Plan 2022–2024 (Decree n. 337, 15 September 2022). The funder was not involved in the study design, collection, analysis, interpretation of data, the writing of this article or the decision to submit it for publication.

Abbreviations

The following abbreviations are used in this manuscript:

2TBN	2-time-slice Temporal Bayesian Network	IED	Intelligent Electronic Device
AEG	Attack Execution Graph	IDS	Intrusion Detection System
AITM	Adversary in the middle	IP	Internet Protocol
ArchG	Architecture Graph	IT	Information Technology
AG	Attack graph	JT	Junction Tree
BF	Brute force	lAG	Local attack graph
BK	Boyen–Koller	MAL	Meta-Attack Language
BN	Bayesian Network	MMS	Manufacturing Message Specification
CA	Certification Authority	NVD	National Vulnerability Database
CPPS	Cyber Physical Power Systems	OPF	Optimal Power Flow
CPT	Conditional Probability Table	OT	Operational Technology
CVSS	Common Vulnerability Scoring System	PKI	Public Key Infrastructure
DAG	Directed Acyclic Graph	PLC	Programmable Logic Controller
DBN	Dynamic Bayesian Network	RV	Random Variable
DDoS	Distributed Denial of Service	SCADA	Supervisory Control And Data Acquisition

DER	Distributed Energy Resources	SGU	Significant Grid User
DMS	DrawNET Modeling System	SLAT	State LookAhead Tree
DMZ	Demilitarized Zone	SQL	Structured Query Language
gAG	General Attack Graph	SSH	Secure Shell
GUI	Graphical User Interface	TLS	Transport Layer Security
HMI	Human Machine Interface	TTC	Time To Compromise
HTTPS	Hypertext Transfer Protocol Secure	UML	Unified Modeling Language
ICS	Industrial Control System	XML	eXtended Markup Language
ICT	Information and Communication Technologies		

References

1. CEI0-16. Norma CEI 0-16:2022-03, Regola Tecnica di Riferimento per la Connessione di Utenti Attivi e Passivi alle reti AT e MT delle Imprese Distributrici di Energia Elettrica. CEI, Milano, Italy. 2022. Available online: https://static.ceinorme.it/strumenti-online/doc/18308.pdf (accessed on 31 July 2024).
2. CEI0-21. Variante V1 della Norma CEI 0-21:2022-03, Regola Tecnica di Riferimento per la Connessione di Utenti Attivi e Passivi alle reti BT delle Imprese Distributrici di Energia Elettrica CEI, Milano, Italy. 2022. Available online: https://static.ceinorme.it/strumenti-online/doc/18066.pdf (accessed on 31 July 2024).
3. ISA/IEC 62443. Standard IEC 62443-4-2:2019, Security for Industrial Automation and Control Systems-Part 4-2: Technical Security Requirements for IACS Components IEC, Geneva, Switzerland, 2019. Available online: https://webstore.iec.ch/en/publication/34421 (accessed on 31 July 2024).
4. Cerotti, D.; Codetta, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Savarro, D.; Terruggia, R. SecuriDN: A Customizable GUI Generating Cybersecurity Models for DER Control Architectures. In Proceedings of the Italian Conference on Cybersecurity, ITASEC 2024, Salerno, Italy, 9–11 April 2024. Available online: http://ceur-ws.org/Vol-3731/ (accessed on 31 July 2024).
5. Wideł, W.; Hacks, S.; Ekstedt, M.; Johnson, P.; Lagerström, R. The meta attack language—A formal description. *Comput. Secur.* **2023**, *130*, 103284. [CrossRef]
6. Cerotti, D.; Codetta-Raiteri, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Terruggia, R. A modular infrastructure for the validation of detection systems. In *Power System Cybersecurity*; Alhelou, H., Hatziargyriou, N., Dongg, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2023; pp. 311–336. [CrossRef]
7. Naval, N.; Yusta, J.M. Virtual power plant models and electricity markets—A review. *Renew. Sustain. Energy Rev.* **2021**, *149*, 111393. [CrossRef]
8. Sarmiento-Vintimilla, J.C.; Torres, E.; Larruskain, D.M.; Pérez-Molina, M.J. Applications, Operational Architectures and Development of Virtual Power Plants as a Strategy to Facilitate the Integration of Distributed Energy Resources. *Energies* **2022**, *15*, 775. [CrossRef]
9. Kabbara, N.; Nait Belaid, M.O.; Gibescu, M.; Camargo, L.R.; Cantenot, J.; Coste, T.; Audebert, V.; Morais, H. Towards Software-Defined Protection, Automation, and Control in Power Systems: Concepts, State of the Art, and Future Challenges. *Energies* **2022**, *15*, 9362. [CrossRef]
10. Docker Inc. Docker. Available online: https://www.docker.com/ (accessed on 31 July 2024).
11. LeMay, E.; Ford, M.D.; Keefe, K.; Sanders, W.H.; Muehrcke, C. Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE). In Proceedings of the 2011 Eighth International Conference on Quantitative Evaluation of SysTems, Aachen, Germany, 5–8 September 2011; pp. 191–200.
12. Rausch, M.J.; Feddersen, B.; Keefe, K.; Sanders, W.H. A Comparison of Different Intrusion Detection Approaches in an Advanced Metering Infrastructure Network Using ADVISE. In *Quantitative Evaluation of Systems, Proceedings of the 13th International Conference, QEST 2016, Quebec City, QC, Canada, 23–25 August 2016*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 279–294. [CrossRef]
13. Keefe, K.; Feddersen, B.; Rausch, M.; Wright, R.; Sanders, W.H. An Ontology Framework for Generating Discrete-Event Stochastic Models. In *Computer Performance Engineering, Proceedings of the 15th European Workshop on Computer Performance Engineering, LNCS 11178, Paris, France, 29–30 October 2018*; Bakhshi, R., Ballarini, P., Barbot, B., Castel-Taleb, H., Remke, A., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2018; pp. 173–189.
14. Keefe, K.; Feddersen, B.; Sanders, W.H.; Muehrcke, C.; Parks, D.; Crapo, A.W.; Gabaldon, A.; Palla, R. Enterprise Security Metrics with the ADVISE Meta Model Formalism. In Proceedings of the International Conference on Emerging Security Information, Systems and Technologies, Venice, Italy, 24–29 August 2015.
15. Mathias, E.; Pontus, J.; Lagerstrom, R.; Gorton, D.; Nydren, J.; Shahzad, K. Securi CAD by Foreseeti: A CAD Tool for Enterprise Cyber Security Management. In Proceedings of the 2015 IEEE 19th Int. Enterprise Distrib. Object Computing Workshop, Adelaide, SA, Australia, 21–25 September 2015; pp. 152–155. [CrossRef]
16. Google LLC. Google Cloud Security Command Cent. Available online: https://cloud.google.com/blog/products/identity-security/introducing-new-capabilities-for-secure-transformations (accessed on 31 July 2024).
17. Ou, X.; Govindavajhala, S.; Appel, A.W. MulVAL: A Logic-based Network Security Analyzer. In Proceedings of the 14th USENIX Security Symposium (USENIX Security 05), Baltimore, MD, USA, 31 July–5 August 2005.

18. Gao, X.; Ali, M.; Sun, W. A Risk Assessment Framework for Cyber-Physical Security in Distribution Grids with Grid-Edge DERs. *Energies* **2024**, *17*, 1587. [CrossRef]
19. Yan, K.; Liu, X.; Lu, Y.; Qin, F. A Cyber-Physical Power System Risk Assessment Model Against Cyberattacks. *IEEE Syst. J.* **2023**, *17*, 2018–2028. [CrossRef]
20. Illinois ADSC CyberSAGE. Available online: https://www.illinois.adsc.com.sg/cybersage/index.html (accessed on 31 July 2024).
21. Temple, W.G.; Wu, Y.; Cheh, C.; Li, Y.; Chen, B.; Kalbarczyk, Z.T.; Sanders, W.H.; Nicol, D.M. CyberSAGE: The cyber security argument graph evaluation tool. *Empir. Softw. Eng.* **2023**, *28*, 18. [CrossRef]
22. George, P.G.; Renjith, V. Evolution of Safety and Security Risk Assessment methodologies towards the use of Bayesian Networks in Process Industries. *Process. Saf. Environ. Prot.* **2021**, *149*, 758–775. [CrossRef]
23. Cheimonidis, P.; Rantos, K. Dynamic Risk Assessment in Cybersecurity: A Systematic Literature Review. *Future Internet* **2023**, *15*, 324. [CrossRef]
24. Cerotti, D.; Codetta, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Terruggia, R. Evidence-Based Analysis of Cyber Attacks to Security Monitored Distributed Energy Resources. *Appl. Sci.* **2020**, *10*, 4725. [CrossRef]
25. Pappaterra, M.J.; Flammini, F. Bayesian Networks for Online Cybersecurity Threat Detection. In *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*; Springer: Cham, Switzerland, 2021; pp. 129–159.
26. SANS E-Isac. Analysis of the Cyber Attack on the Ukrainian Power Grid. 2016. Available online: https://nsarchive.gwu.edu/sites/default/files/documents/3891751/SANS-and-Electricity-Information-Sharing-and.pdf (accessed on 31 July 2024).
27. Slowik, J. CRASHOVERRIDE Malware. 2018. Available online: https://www.dragos.com/wp-content/uploads/CRASHOVERRIDE2018.pdf (accessed on 31 July 2024).
28. Stouffer, K.; Pease, M.; Tang, C.; Zimmerman, T.; Pillitteri, V.; Lightman, S.; Hahn, A.; Saravia, S.; Sherule, A.; Thompson, M. *Guide to Operational Technology (OT) Security*; Technical Report SP 800-82 Rev 3, NIST; US Department of Commerce, National Institute of Standards and Technology: Gaithersburg, MD, USA, 2023.
29. *IEC 61850:2024 SER*; IEC TC 57—Power Systems Management and Associated Information Exchange. IEC: Geneva, Switzerland, 2024.
30. *IEC 62351:2024 SER*; IEC TC 57—Power Systems Management and Associated Information Exchange. IEC: Geneva, Switzerland, 2024.
31. The MITRE Corporation. Adversarial Tactics, Techniques and Common Knowledge (ATT&CK). 2015. Available online: https://attack.mitre.org/ (accessed on 31 July 2024).
32. The MITRE Corporation. ATT&CK for Enterprise. 2015. Available online: https://attack.mitre.org/matrices/enterprise/ (accessed on 31 July 2024).
33. The MITRE Corporation. ATT&CK for Industrial Control Systems. 2020. Available online: https://attack.mitre.org/matrices/ics/ (accessed on 31 July 2024).
34. The MITRE Corporation. Cyber Analytics Repository (CAR). Available online: https://car.mitre.org/wiki/Main_Page (accessed on 31 July 2024).
35. Chockalingam, S.; Pieters, W.; Teixeira, A.; van Gelder, P. Bayesian Network Models in Cyber Security: A Systematic Review. In *Secure IT Systems, Proceedings of the NordSec 2017, Tartu, Estonia, 8–10 November 2017*; Lipmaa, H., Mitrokotsa, A., Matulevičius, R., Eds.; Springer: Cham, Switzerland, 2017; pp. 105–122.
36. Misuri, A.; Khakzad, N.; Reniers, G.; Cozzani, V. A Bayesian network methodology for optimal security management of critical infrastructures. *Reliab. Eng. Syst. Saf.* **2019**, *191*, 106112. [CrossRef]
37. Murphy, K. Dynamic Bayesian Networks: Representation, Inference and Learning. Ph.D. Thesis, University of California, Berkeley, CA, USA, 2002.
38. Kiaerulff, U. dHugin: A computational system for dynamic time-sliced Bayesian networks. *Int. J. Forecast.* **1995**, *11*, 89–111. [CrossRef]
39. Pamfil, R.; Sriwattanaworachai, N.; Desai, S.; Pilgerstorfer, P.; Georgatzis, K.; Beaumont, P.; Aragam, B. DYNOTEARS: Structure Learning from Time-Series Data. In Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics, Online, 26–28 August 2020; Chiappa, S., Calandra, R., Eds.; Proceedings of Machine Learning Research: Cambridge, MA, USA, 2020; Volume 108, pp. 1595–1605. Available online: https://proceedings.mlr.press/v108/pamfil20a.html (accessed on 31 July 2024).
40. Huang, C.; Darwiche, A. Inference in belief networks: A procedural guide. *Int. J. Approx. Reason.* **1996**, *15*, 225–263. [CrossRef]
41. Boyen, X.; Koller, D. Tractable Inference for Complex Stochastic Processes. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, Madison, WI, USA, 24–26 July 1998; pp. 33–42.
42. Pearl, J. *Probabilistic Reasoning in Intelligent Systems*; Morgan Kaufmann: Burlington, MA, USA, 1989.
43. Murphy, K.; Russell, S. Rao-blackwellised particle filtering for dynamic Bayesian networks. In *Sequential MOnte-Carlo Methods in Practice*; Springer: Berlin/Heidelberg, Germany, 2001.
44. Docker Inc. Docker Swarm. Available online: https://docs.docker.com/engine/swarm/ (accessed on 31 July 2024).
45. Rapid7. Metasploit. Available online: https://www.metasploit.com/ (accessed on 31 July 2024).
46. Boelen, M. Audit. Available online: https://linux-audit.com/ (accessed on 31 July 2024).
47. Elasticsearch, B.V. Auditbeat. Available online: https://www.elastic.co/beats/auditbeat (accessed on 31 July 2024).
48. Django Software Foundation. OpenSearch. Available online: https://opensearch.org/ (accessed on 31 July 2024).

49. Apache Software Foundation. Kafka. Available online: https://kafka.apache.org/ (accessed on 31 July 2024).
50. Codetta-Raiteri, D.; Franceschinis, G.; Gribaudo, M. Defining formalisms and models in the Draw-Net Modelling System. In Proceedings of the International Workshop on Modelling of Objects, Components and Agents, Turku, Finland, 26 June 2006; pp. 123–144.
51. Codetta-Raiteri, D.; Portinale, L. A Petri net-based tool for the analysis of generalized continuous time Bayesian networks. In *Theory and Application of Multi-Formalism Modeling*; IGI Global: Pennsylvania, PA, USA, 2013; pp. 118–143.
52. Beccuti, M.; Codetta-Raiteri, D.; Franceschinis, G.; Haddad, S. Non deterministic Repairable Fault Trees for computing optimal repair strategy. In Proceedings of the International Conference on Performance Evaluation, Methodologies and Tools, Athens, Greece, 20–24 October 2008.
53. NIST. CVE-2019-14271. Available online: https://nvd.nist.gov/vuln/detail/CVE-2019-14271 (accessed on 31 July 2024).
54. Avrahami, Y. CVE-2019-14271 Article. Available online: https://unit42.paloaltonetworks.com/docker-patched-the-most-severe-copy-vulnerability-to-date-with-cve-2019-14271/ (accessed on 31 July 2024).
55. Roethlisberger, D. SSLsplit. Available online: https://www.roe.ch/SSLsplit (accessed on 31 July 2024).
56. Tayouri, D.; Baum, N.; Shabtai, A.; Puzis, R. A Survey of MulVAL Extensions and Their Attack Scenarios Coverage. *IEEE Access* **2023**, *11*, 27974–27991. [CrossRef]
57. Sabur, A.; Chowdhary, A.; Huang, D.; Alshamrani, A. Toward scalable graph-based security analysis for cloud networks. *Comput. Netw.* **2022**, *206*, 108795. [CrossRef]
58. Gribaudo, M. DrawNET 4. Available online: https://www.draw-net.com/ (accessed on 31 July 2024).
59. Eaton, J.W. Octave. Available online: https://www.gnu.org/software/octave/ (accessed on 31 July 2024).