DiSIT, Computer Science Institute
Università del Piemonte Orientale “A. Avogadro”
Viale Teresa Michel 11, 15121 Alessandria
http://www.di.unipmn.it

UNIVERSITÀ DEL PIEMONTE ORIENTALE

SERICS Report on Dynamic Bayesian Network and Generalized Stochastic Petri Net model

E. Amparore, M. Borrelli, D. Cerotti, S. Donatelli, G. Franceschinis, D. Savarro (inserire indirizzo e-mail, inserire indirizzo e-mail, davide.cerotti@uniupo.it, susi@di.unito.it, giuliana.franceschinis@uniupo.it, davide.savarro@uniupo.it)

TECHNICAL REPORT TR-INF-2026-06-01-UNIPMN
(June 2026)

Research Technical Reports published by DiSIT, Computer Science Institute, Università del Piemonte Orientale are available via WWW at URL `http://www.di.unipmn.it/`. Plain-text abstracts organized by year are available in the directory

Recent Titles from the TR-INF-UNIPMN Technical Report Series

2024-01 *Temporal Many-valued Conditional Logics: a Preliminary Report*, M. Alviano, L. Giordano, D. Theseider Dupré, September 2024.

2023-02 *Sviluppo in OMNeT++ di modelli di cyber attacchi a risorse energetiche distribuite*, D. Savarro, December 2023.

2023-01 *NRTS: A Client-Server architecture for supporting data recording, transmission and evaluation of multidisciplinary teams during the neonatal resuscitation simulation scenario*, M. Canonico, S. Montani, M. Striani, April 2023.

2022-01 *A modular infrastructure for the validation of cyberattack detection systems*, D. Cerotti, D. Codetta Raiteri, G. Dondossola, L. Egidi, G. Franceschinis, L. Portinale, R. Terruggia, May 2022.

2021-01 *General composition for Symmetric Net arc functions with applications*, L. Capra, M. De Pierro, G. Franceschinis, May 2021.

2020-05 *Weighted defeasible knowledge bases and a multipreference semantics for a deep neural network model*, L. Giordano, D. Theseider Dupré, December 2020.

2020-04 *A reconstruction of multipreference closure*, L. Giordano, V. Gliozzi, September 2020.

2020-03 *A framework for a modular multi-concept lexicographic closure semantics*, L. Giordano, D. Theseider Dupré, September 2020.

2020-02 *On a plausible concept-wise multipreference semantics and its relations with self-organising maps*, L. Giordano, V. Gliozzi, D. Theseider Dupré, September 2020.

2020-01 *Reasoning about exceptions in ontologies: from the lexicographic closure to the skeptical closure*, L. Giordano, V. Gliozzi, March 2020.

2019-05 *Renvoi in Private International Law: a Formalization with Modal Contexts*, L. Giordano, B. Matteo, K. Satoh, October 2019.

2019-04 *UML class diagrams supporting formalism definition in the Draw-Net Modeling System*, D. Codetta Raiteri, July 2019.

2019-03 *Tracing and preventing sharing and mutation*, P. Giannini, M. Servetto, E. Zucca, July 2019.

2019-02 *The Android Forensics Automator (AnForA): a tool for the Automated Forensic Analysis of Android Applications*, C. Anglano, M. Canonico, M. Guazzone, June 2019.

2019-01 *Deriving Symbolic and Parametric Structural Relations in Symmetric Nets: Focus on Composition Operator*, L. Capra, M. De Pierro, G. Franceschinis, March 2019.

2018-03 *Deriving Symbolic Ordinary Differential Equations from Stochastic Symmetric Nets without Unfolding*, M. Beccuti, L. Capra, M. De Pierro, G. Franceschinis, S. Pernice, July 2018.

SERICS Report on Dynamic Bayesian Network and Generalized Stochastic Petri Net models (Q-CPS^2 - Work Package 3)

Davide Savarro[1], Mattia Borrelli[1], Elvio Gilberto Amparore[1], Davide Cerotti[2], Giuliana Franceschinis[2], and Susanna Donatelli[1]

[1]Computer Science Department, Università di Torino, Torino, Italy
{davide.savarro, mattia.borrelli, elviogilberto.amparore, susanna.donatelli}@unito.it
[2]Computer Science Institute, DiSIT, Università del Piemonte Orientale, Alessandria, Italy , {davide.cerotti, giuliana.franceschinis}@uniupo.it

Contents

1 Introduction

In recent years, the progressive digitization of electrical distribution networks has created an unprecedented bond between cyber infrastructure and physical assets. While this integration introduces new means by which monitoring and control operations can be performed, it also expands the attack surface and enables multi-stage intrusions whose effects propagate from compromised IT/OT devices to the power system itself. In this context, security assessment methodologies must reason simultaneously on how an attacker can advance in a plausible cyber kill chain and how his resulting actions can manifest on grid topology and power system measurements in general. A striking example of such cybertheats can be found the Industroyer 1 attack campaign [37], where unauthorized command injection tripping critical circuit breakers caused an extensive blackout in Ukraines's capital Kyiv in December 2016.

This work, according to the main objectives of Work-Package 3 of the Q-CPS^2 project[1], presents a security assessment modeling and analysis pipeline that starts from a high-level Attack Graph (AG) of an Industroyer-like campaign and systematically derives two complementary inference models: a Dynamic Bayesian Network (DBN) and a Generalized Stochastic Petri Net (GSPN). The DBN captures the temporal dependencies among attack-steps and power-system measurements, allowing to perform inference tasks under uncertain conditions. The GSPN encodes the same process modeling the explicit competition between attack progression and mitigation, supporting temporal probabilistic model checking. We estimate attack-step completion/initiation time and use them via a continuous-to-discrete approximation to parameterize the CPTs of attack-step related nodes of the DBN and GSPN rates/weights which encode the same semantics. To ground the analysis and train the power-related sections of our models, we leverage Wattson, a co-simulation framework that emulates IEC-104 SCADA communications and power-flow dynamics. We extend it with wautorunner, a module that generates labeled traces for both nominal and under-attack scenarios across diverse load/generation profiles and attacker strategies. Finally we perform a comparative analysis between the type of queries that DBNs and GSPNs can answer when applied on a benchmark simulated scenario (CIGRE MV all-DER).

The remaining part of this work is structured ad follows. Section 2 introduces preliminary concept such as Wattson framework / IEC-104, MITRE ATT&CK framework, AGs, DBNs and GSPNs. Section 3 details the Wattson and AG reference models and the AG-to-DBN/GSPN mapping. Section 4 describes wautorunner and the traces generation process. Section 5 presents the parameterization strategies for both DBN and GSPN. Section 6 reports the inference results of both models in multiple attack scenarios and for different queries respectively. Finally Section 7 discusses related work and Section 8 concludes the work highlighting possible limitations and future directions.

2 Preliminary concepts

In this section a series of key concepts and definitions are presented in order to lay a background knowledge for the next sections.

[1]The Quantitative models for Cyber Physical Systems Security (Q-CPS^2) project is part of the Spoke n. 8 of the SERICS project.

2.1 Wattson and the IEC-104 standard

Without the possibility of utilizing real-world cyberthreat intelligence, the first step of our analysis focused on identifying a trustworthy and easily accessible data source to obtain the information required to train our inference models. Wattson [4] is a co-simulation framework that facilitates the implementation and the analysis of cyberattacks targeting power grids, allowing to reproduce both their fingerprint on ICT communications and their impact on physical devices. It builds on Containernet [34] for network communication emulation and PandaPower [45] for power grid simulation, coordinated by a synchronization layer to allow realistic and scalable representation of monitoring and control networks of Cyber Physical Power Systems (CPPSs). Wattson reproduces IEC 60870-5-104 [14] communications between a Master Terminal Unit (MTU) and multiple Remote Terminal Units (RTUs) serving as Intelligent Electronic Devices (IEDs) attached to one or multiple grid components. Those components transmit monitoring information, such as voltage and current measurements, to the MTU which can issue control commands to explicitly monitor or actively manage the grid. In an attack scenario such as Industroyer 1 and 2, the attacker interrupts the normal communication and repeatedly sends single commands (identified by the IEC 104 protocol as C_SC_NA_1) or double commands (C_DC_NA_1) to RTUs controlling crytical power components such as switches and circuit breaker to disrupt the power flow. Thanks to this emulation environment we can reproduce such anomalous topology changes without the need for a physical testbed that could be damaged by such malicious actions. Other than practical research tool that allow to reproduce attack scenarios, it is crucial to adopt a well established attack ontology to adequately identify and isolate each step required by the attacker to pursue its goal.

2.2 MITRE ATT&CK Framework

Leveraging the already existing knowledge about real-world cyberattacks and maintaining a uniform terminology with the existing research, led us to choose the MITRE ATT&CK framework [41]. This comprehensive and constantly updated ontology is widely used by researchers and organizations to systematically categorize, analyze and mitigate cyber threats. In our specific use case we combined the information presented in reports describing the Industroyer 1 cyberattack, such as [36], with the description of the ukranian ICS cyberattacks included in campaigns [39] and [40] of the ATT&CK framework. From these sources, we identified the main techniques employed during the 2016 events and integrated with additional techniques from the Enterprise Matrix [43] and ICS Matrix [44] of the ATT&CK framework. The first matrix focuses on attack methodologies related to desktop and cloud environments, while the second one is specialized on cyber threats directly related to Industrial Control Systems. Most of the analytics are instead modeled based on the MITRE ATT&CK Data Sources [42], that represent the various subjects/topics of information that can be collected by sensors/logs and that can be relevant to detect a given ATT&CK technique or sub-technique. Our goal was to develop a comprehensive and up-to-date representation of all steps leading to the final power system disruption, modeling their dependencies with a structured approach.

2.3 Attack Graphs

AGs are a semi-formal way of describing attack processes as sequences of steps from one initial condition to a target goal, or more broadly from multiple initial states to multiple

goals. Specifically, we define every directed path in the graph from an initial state to a final goal as a possible successful attack process. Many variation of AGs have been proposed in the literature, but we choose to maintain a representation we already used on one of our previous works [9]. In this use-case it was necessary to introduce just the following types of node:

- **Technique nodes**: represent individual attack techniques derived from the ATT&CK matrices.
- **Logical nodes**: model logical operations combining two or more technique nodes. A Logical AND node is active if and only if all its parent technique nodes are active, whereas a logical OR node is active if at least one of its parent technique nodes is active.
- **Analytic nodes**: represent an event that could inform the security analyst about the completion of an attack step.

Based on the constraints of our formalism a directed arc can connect different types of nodes in the AG, with different meanings:

- Technique node $\longrightarrow$ Technique node: meaning that the first technique enables the second one.
- Technique node $\longrightarrow$ Logical node: models the combination of two or more techniques.
- Logical node $\longrightarrow$ Technique node: the technique is enabled by a combination of multiple techniques.
- Technique node $\longrightarrow$ Analytic node: the execution of the technique influences the generation of the associated analytic.

This model serves as an intermediate attack modeling framework towards the generation of inference models capable of answering a wide variety of queries.

2.4 Dynamic Bayesian Networks

Bayesian Networks (BNs) have been one of the most adopted formalisms in the area of probabilistic and causal inference. In recent years, they have also proven to be a valuable tool for addressing security assessment problems [13, 27, 8], especially related to the area of cyberattack explainability and evidence correlation. A BN is defined as a pair $N = \langle\langle V, E\rangle, P\rangle$, where $\langle V, E\rangle$ are nodes and edges of a Directed Acyclic Graph (DAG) and P is a probability distribution over V. Each node of the network $V = \{X_1, ..., X_n\}$ is associated to a Random Variable (RV) that expresses the a probabilistic relationship 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). When considering just discrete RVs, each local distribution can be specified in the form called Conditional Probability Table (CPT). In this data structure each column expresses a combination of states of the parent variables in the DAG and each row is associated to a different state of the current variable. Each cell contains the conditional probability of the current state for that row, given the combination of states of parent variables, expressed in that column. To model more complex time-dependent phenomena, DBNs [28, 21] introduce an explicit temporal dimension splitting the model into multiple time slices. We choose a

specific version of DBN called 2-time-slice Temporal Bayesian Network (2TBN), where the set of nodes V is replicated over two consecutive time slices, t and t'. With this constraints it is possible to define two types of edges:

- *intra-slice* edges: an edge from a node X_i^t to a node X_j^t, with $i \neq j$. These are the same one we define in classic BNs.
- *inter-slice* edges (1^{st} order): an edge from a node X_i^t to a node $X_j^{t'}$, where i can be equal to j. These edges express the temporal dependencies between nodes in different time slices. They model in which way a node at time slice t' is influenced by another node or by itself at time slice t.

It is worth noting that while DBNs are a *discrete* time model and we need them to approximate a *continuous* time reality. We will further discuss this aspect in Section 5.1.

2.4.1 Inference tasks and algorithms

While classic BNs can just perform predictive and diagnostic inference on a single static snapshot of the model, DBNs can execute a wider variety of inference tasks depending on how unobserved variables (e.g. associated to specific attack steps) and observed variables (e.g. related to analytics raising alerts) are placed within the timeline. The following inference tasks are possible:

- **Filtering**: computing the likelihood of a state at time t (now) given the evidence available up to the current time slice. From a cybersecurity standpoint, this strategy can be adopted to monitor the current state of potential threats based on real time information.
- **Prediction**: computing the probability of a future state at time $t + h$ (with $h > 0$) given the evidence available up to the current time slice t. This task could solve the problem of predicting a future state of a cyberattack based on currently known information.
- **Smoothing**: computing the probability of an outcome at a generic time step k given any previous and subsequent evidence (e.g. at time $k - l$ and $k + j$ with $l, j > 0$).

All these types of inference tasks can be performed by a wide variety of inference algorithms. An exact algorithm for DBN inference is the 1.5JT [28], which converts the DBN into a *Junction Tree* (JT), on which the inference is performed. The applicability of an exact computation depends on the number of nodes in the network, the average number of outcomes for each node, the average degree of the network and by the chosen discretization step (defines the granularity of the analysis). The *Boyen-Koller* algorithm (BK) [6] can be considered an approximate version of the 1.5JT. BK is still based on the JT data structure, but the clusters of nodes in the DBN that generate macro-nodes in the JT, are defined by the user. The level of approximation depends on how variables are distributed in clusters; this allows to tune the algorithm to obtain from an exact computation (a single cluster contains al the variables) to a fully factorized one (one variable per cluster returning the least precise results) and all intermediate variants in between. The last category of approximate algorithms, such as *likelihood weighting* [33], *particle filtering* [16] and Adaptive Importance Sampling (AIS) [11], exploit stochastic simulation. In our experiments, based on the capabilities of the SMILE [5] framework, we adopted a modified version of the Evidence

Pre-propagation Importance Sampling (EPIS) algorithm [48], to perform filtering inference. The EPIS algorithm uses loopy belief propagation to compute an estimate of the posterior probability over all nodes of the network and then uses importance sampling to refine this estimate. Other than being the most accurate of all the sampling algorithms (even when providing rare evidence streams), it is even the fastest, avoiding the costly learning phase of the AIS algorithm.

Independently by the chosen inference algorithm used, DBNs can only answer probabilistic queries regarding the state of unobserved variables in specific points in time, given some evidence (e.g. "what's the probability that an attack step A is completed by time slice 40 given some evidence B, C and D in the previous slices?"). Each type of inference task involves conditioning the model based on a single "trajectory" of evidence and then computing the belief state at one or more points in time of that same trajectory. While useful, there are limits to the expressive power of queries in DBN, in fact we could be asking what's the probability that a specific attack step A is completed before an attack step B. This require of computing the set of trajectories satisfying the statement (where A raises before B) and computing their likelihood based on all possible trajectories. Due to these limitations we decided to explore other modeling formalisms capable of answering such queries thanks to the possibility of applying Model Checking algorithms on their definition.

2.5 Generalized Stochastic Petri Nets

The alternative model is the Generalized Stochastic Petri Net (GSPN) one. GSPNs[1][25] are a modeling formalism that can be used for the analysis of complex models of dynamic systems and for the evaluation of their performance and reliability. Modelling with GSPN allows to naturally study events paths, representing both the probabilistic aspect and the sequentiality of the events, including a time-based competition relationships between states, which can be very useful in cyber threats scenarios. This kind of net, based on well-known Petri nets formalism, is defined as an 8-tuple $GSPN = (P, T, pri, I, O, H, m_0, W)$ where:

- SPN $= (P, T, pri, I, O, H, m_0)$ is the underlying Stochastic Petri Net with priority, a by-partite directed graph where:
  - P is the set of places.
  - T is the set of transitions. T can be partitioned in two subset of *immediate* and *exponential* (or *timed*) transitions respectively.
  - $pri : T \to \mathbb{N}$ is the priority function where timed transitions are always associated with priority zero, while immediate transitions must have priority greater than zero.
  - I, O and H are the functions which describe input, output and inhibitor arcs in the net.
  - m_0 is the initial marking which also defines the beginning state of the net.
- $W : T \to \mathbb{R}$ is a weigth function which associates real positive values to all the transitions. The weights semantics depends on transition type.

Similarly to DBNs, GSPN models keep part of the probabilistic component in the weight value of the immediate transition, which however represents the probability used for the

resolution of conflicts, allowing us to model the concept of immediate choice between different paths. Then, a GSPN includes temporal evolution, which can be found in the weight of exponential transitions. Indeed, each timed transition t is associated to a *rate* which represents the distribution of the delay of t itself. The GSPN usage lead us to a step-by-step modeling of the attack under examination, including, for each step, both the (probabilistic) triggering of the relevant evidence representing the identification of the attack completed at a certain step, and a conflict between the possible mitigation of the attack at a certain step or the completion of the next step. This conflict depends on the respective completion times.

2.5.1 Statistical Model Checking and verification

A Petri Nets execution generates different paths of events, according to the net structure. Then, a preliminary analysis could be to check the truthfulness of path-dependent properties; for instance: "it is true that all the executions of the Industroyer attack net end with a power system failure and the loss of control on the SCADA control room". This kind of model checking can be performed on PN (and GSPN as well) by using temporal logic, such as LTL or CTL, to express properties.

However, similar queries could be extended with the concepts of time passing and probabilty which are included in GSPN by definition; then the analysis focuses on more expressive **temporal probabilistic properties** such as "with probabilty greater than 0.75 Industroyer attack will cause a system power failure and the loss of control on SCADA control room between 30 and 60 seconds from the beginning of the attack." A stochastic logic which allows to represents properties like the example above is CSL^{TA},[15] because is capable to express both temporal path-dependent and probabilistic properties by using Single Clock *Deterministic Timed Automata* (DTA) for specification. This is the main addition comparing CSL^{TA} to its base language, CSL: instead of using common operators in temporal logic like *Until* or *Next*, path-formulas are specified with DTA, which are tools that try to simulate the model behavior. A DTA $\mathcal{A}$ is a tool defined as $\mathcal{A} = \langle \sum, Act, L, \triangle, Init, Final, \rightarrow \rangle$ where $\sum, Act$ and L are finite sets of state propositions, actions and locations respectively; $\triangle$ is a labeling function over L which defines properties that must be true on the locations (the set of label associated to $l \in L$ is $Lab(l)$); $Init$ and $Final$ are subsets of L each containing initial and final states of the automaton. Last, given a single clock x defined as a variable whose value increases linearly with time, for single clock DTA $\rightarrow$ is the set of edges defined as:

$$\rightarrow \subseteq L \times ((InnerEdge \times 2^{Act} \cup (BoundaryEdge \times \{\#\})) \times \{\varnothing, x\} \times L$$

Tagging an edge with x equals to a reset of the x value. In sigle clock DTA, edges can be distinguished based on the constraint type, the actions associated and the trigger logic:

- **Inner Edges** have constraint in the form of $\alpha < x < \beta$ with $\alpha, \beta \in \mathbb{R}_{\geq 0}$ and a set of allowed actions defined in the model. Inner edges are triggered by model transitions.
- **Boundary Edges** have constraint in the form of $x = \alpha$ and no action associated (the special symbol # is used). Boundary edges are triggered by the elapse of time. Besides, they are urgent and have priority over transitions and Inner edges trigger.

Given the complete single clock DTA definition and $\lambda \in [0,1]$ as a probability, $p \in AP$ as an atomic proposition and $\bowtie \in \{\leq, <, >, \geq\}$ as a comparison operator, a CSL^{TA} state formula Φ is defined as

$$\Phi ::= p|\neg\Phi|\Phi \wedge \Phi|S_{\bowtie\lambda}(\Phi)|P_{\bowtie\lambda}(\mathcal{A})$$

The semantics of CSL^{TA} in a state $s \in S$ is defined by:

$$\mathcal{M}, s \vDash p \Leftrightarrow p \in lab(S)$$

$$\mathcal{M}, s \vDash \neg\Phi \Leftrightarrow \mathcal{M}, s \nvDash \Phi$$

$$\mathcal{M}, s \vDash \Phi_1 \wedge \Phi_2 \Leftrightarrow \mathcal{M}, s \vDash \Phi_1 \wedge \mathcal{M}, s \vDash \Phi_2$$

$$\mathcal{M}, s \vDash S_{\bowtie\lambda}(\Phi) \Leftrightarrow \sum_{s' \in S \wedge \mathcal{M}, s' \vDash \Phi} \pi(s, s') \bowtie \lambda$$

$$\mathcal{M}, s \vDash P_{\bowtie\lambda}(\mathcal{A}) \Leftrightarrow Pr_s^{\mathcal{M}}(AccPath^{\mathcal{M}}(s, \mathcal{A})) \bowtie \lambda$$

with $\pi(s, s')$ the steady-state probability of state s', starting in state s, $\mathcal{A}$ an acceptance automaton, and $AccPath^{\mathcal{M}}(s, \mathcal{A})$ the paths of $\mathcal{M}$ that starts in s and are accepted by $\mathcal{A}$.

Originally, CSL^{TA} has been proposed to perform model checking only on a model $\mathcal{M}$ in the form of Continuous time Markov chains (CTMC) [15]. However, a GSPN can be always converted directly into an associated stochastic process, which is a CTMC as well, if its reachability graph is finite (as it will be for the GSPN presented in Section 3.4). Thus, CSL^{TA} model checking is also possible by adopting a GSPN as $\mathcal{M}$. A solid and efficient algorithm to solve model checking with CSL^{TA} has already been proposed,[3] and there are implementations in tools such as GreatSPN[12] which includes $MC4CSL^{TA}$ model checker.[2] So, for our experiments we used GreatSPN both to build GSPN and to perform model checking, including the DTAs specifications representing different queries.

3 Reference Models

In this section all the models developed throwout the project period will be presented.

3.1 Wattson model

As already mentioned in Section 2.1 the Wattson framework is capable of reproducing both the ICT monitoring and control infrastructure and the operated physical devices within the simulated power grid. In our use case we decided to keep Wattson's reference scenario which is based on a well established model: the *CIGRE MV all DER* power grid topology, provided by pandapower [30] and based on the CIGRE Task Force C6.04.02 paper [38].

This scenario is composed by 12 Medium Voltage (MV) Distribution Substations (DSSs) connected to a High Voltage to Medium Voltage (HV/MV) Transmission Substation (TSS). The power grid model, whose line diagram is shown in Figure 1, contains a total of 2 HV/MV transformers, 15 busses, 15 lines, 18 loads, 13 generators (including photovoltaic generators, wind turbines, residential fuel cells and CHP diesel generators), 2 storage components and 8 circuit switches. Notice how the power grid is split into two feeders (1 and 2) and we will use this information to build a more informative DBN as we will see in Section 3.3. By default all generators and loads operate at a static power infeed or consumption and the initial circuit

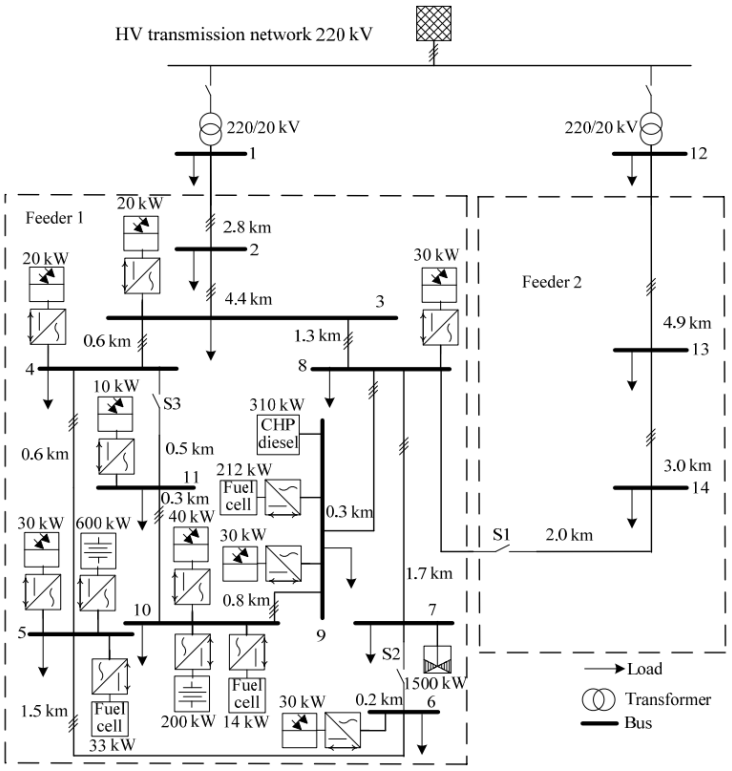


Figure 1: *CIGRE MV all DER* power grid model

breakers configuration is always the same, but we will see in Section 4.1 how this initial configuration is modified to produce different variations of the same scenario. Focusing on the ICT infrastructure, a total of 32 RTU, divided into two Operational Technology (OT) subnetworks, are controlled by a single MTU via IEC 104 protocol.

3.2 Attack Graph model

The cyber kill chain of the Industroyer 1 cyberattack consists of a series of tightly coupled techniques executed by the attacker to reach the goal of disrupting the target power grid. While most studies on this malware tend to focus on the power system related section of the attack, when the attacker is already capable of sending malicious control commands, it is equally important to consider the valuable insight we could gather from the earlier stages, including the initial actions performed on the compromised host and the post-exploitation activities that may follow. Figure 2 shows the AG that models such techniques and their interdependencies, based on the information extracted from the data sources presented in Section 2.2.

The circular nodes represent the techniques executed by the attacker during the Industroyer 1 attack, while the notebook-shaped nodes denote the analytics associated to the corresponding technique nodes. Each attack step is mapped to the corresponding technique in the ATT&CK matrix and the tactic it implements. In those cases where a single technique can be used for various tactics, we choose according to the objective the attacker plans to

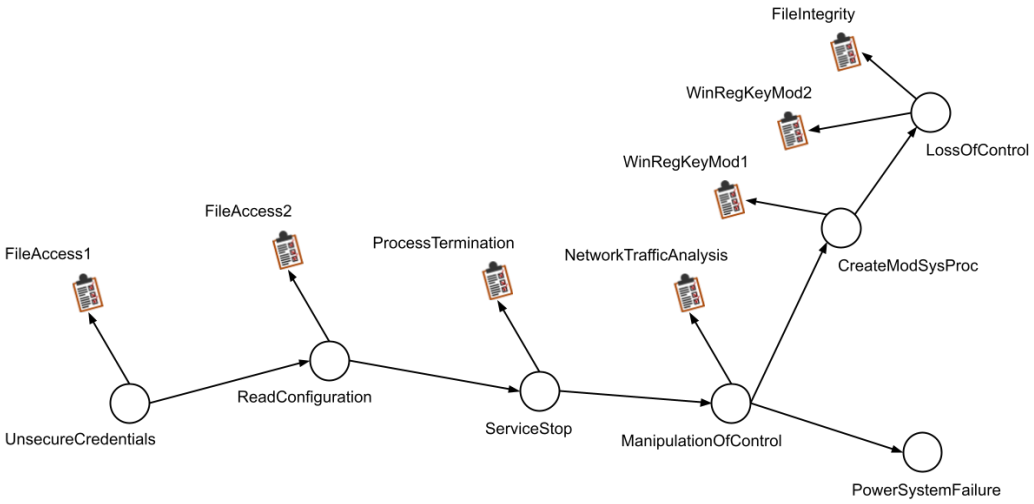


Figure 2: Industroyer 1 Attack Graph

achieve. Where the value "NA" replaces the matrix, tactic or technique name, it indicates that the step is not explicitly mentioned in the MITRE Industroyer campaigns but has been inferred from analysis reports such as [36]. The following attacks are included in the model.

- **Private Key theft** (Enterprise ATT&CK matrix—Tactic: Credential Access—Technique: Unsecured Credentials): The attacker obtain an insecurely stored private key and the corresponding certificate. This allows him to access the target host, which in out scenario runs the SCADA control center within the MTU component.
- **Read Configuration file** (NA—Tactic: NA—Technique: NA): The malware read the configuration file to gather information about the targets and the attack strategy to adopt.
- **IEC-104 Client Stop** (Enterprise ATT&CK matrix—Tactic: Impact— Technique: Service Stop): The Industroyer malware blocks the original IEC-104 client communications with the RTUs and start new connections to allow remote monitoring and control of such devices.
- **Malicious Command Injection** (ICS ATT&CK matrix—Tactic: Impact—Technique: Manipulation of Control): The malware starts issuing malicious control command to circuit breakers to alter their state and cause sudden changes in power network topology.
- **System Process Alteration** (Enterprise ATT&CK matrix—Tactic: Persistence—Technique: Create or Modify System Process: Windows Service): The malware create or modify Windows system services to acquire persistence even after reboot.
- **Loss of Control** (ICS ATT&CK matrix—Tactic: Impact—Technique: Loss of Control): Industroyer's data wiper component deletes the configuration files of the control center and removes the registry image path throughout the system, rendering the host unusable and the following recovery process more difficult.

- **Power System Failure** (NA—Tactic: NA—Technique: NA): Represents a generic final objective of the attacker, corresponding to physical damage inflicted on the power grid.

Attack Steps	**(Abbreviations)**	**Analytics**	**(Abbreviations)**
Private Key theft	(*UnsecureCredentials*)	access to files containing credentials	(`FileAccess1`)
Read Configuration file	(*ReadConfiguration*)	access to suspicious files	(`FileAccess2`)
IEC-104 Client Stop	(*ServiceStop*)	termination of system process	(`ProcessTermination`)
Malicious Command Injection	(*ManipulationOfControl*)	presence of abnormal traffic patterns	(`NetworkTrafficAnalysis`)
System Process Alteration	(*CreateModSysProc*)	modification of registry key	(`WinRegKeyMod1`)
Loss of Control	(*LossOfControl*)	modification of registry key	(`WinRegKeyMod2`)
		deletion of crytical configuration files	(`FileIntegrity`)
Power System Failure	(*PowerSystemFailure*)		

Table 1: Attack steps and analytics

Some of these attack steps can be exposed by an implemented analytic. Table 1 summarizes Figure 2, mapping each attack step to its corresponding abbreviation and associated analytic(s), for which a brief description is provided. From the AG presented above, two separate inference models where derived, a DBN and a GSPN, each designed to address different types of queries using complementary approaches.

3.3 Dynamic Bayesian Network model

The DNB derived from the AG presented in Section 3.2 consists of two main components. The first component map almost directly onto the nodes of the AG that model the Industroyer 1 attack steps and is shown in Figure 3. The second component, presented in Figure 4, extends the *PowerSystemFailure* node of the AG by modeling power grid related features that are a reliable predictors of the power system operational conditions.

3.3.1 Modeling the Attack Graph

Focusing on the first component, this section of the DBN can be directly mapped to the AG previously presented. All technique nodes are converted into temporal nodes within the DBN by adding self temporal arcs that capture its evolution over time. Each temporal node has two possible outcomes: one representing the state in which the attack step has not yet been completed (denoted as `U` for *Uncompleted*) and the other representing the successful completion of the technique (denoted as `C` for *Completed*). The only exception is the *ManipulationOfControl* node which, due to its multi-strategy nature, includes a single outcome representing the state in which the attack step has not yet started (denoted as `N` for *Not started*) and multiple additional outcomes, one for each strategy that the attacker may choose to execute (*Alternate*, *OpenOnly*, *CloseOnly* and *Explicit*). This aspect is examined in greater detail in Section 4.

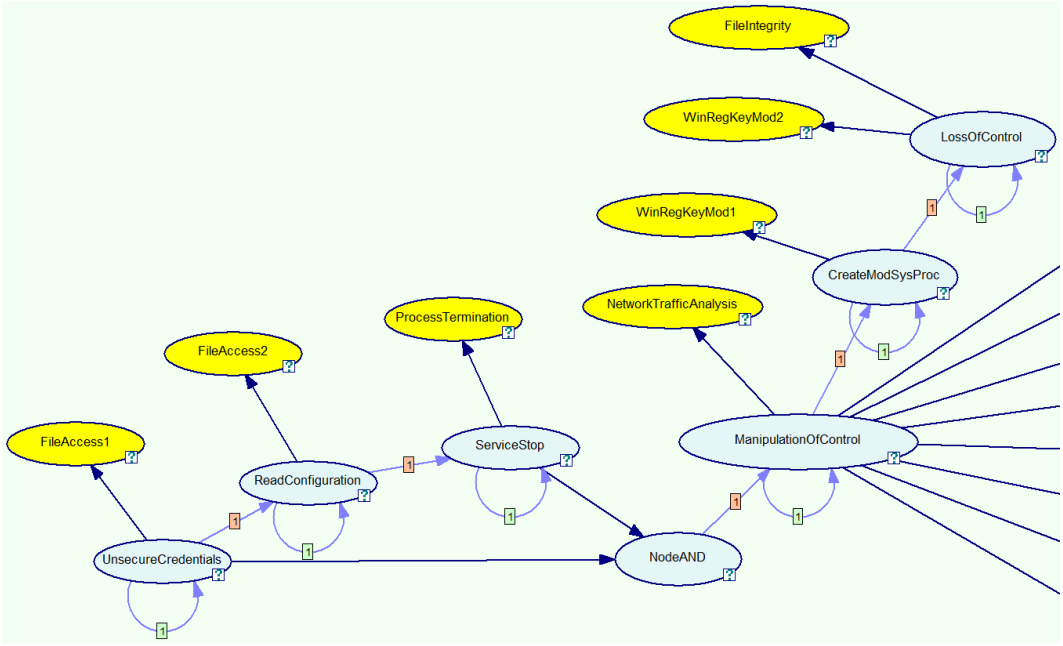


Figure 3: First component of the DBN modeling the Industroyer 1 cyber kill chain

Another notable difference from the initial AG is the inclusion of an AND logical node (*NodeAND*) that models the combination of its preconditions (*UnsecureCredentials* and *ServiceStop*). In this case, the *ManipulationOfControl* attack step requires the combination of the legitimate IEC-104 client to be stopped (*ServiceStopped*) and a constant ssh connection acquired after private key theft (*UnsecureCredentials*) to stay active. This integration necessary to correctly combine preconditions that can be either persistent or not persistent, but we will come back on this concept in Section 5.1, where we will need it to parameterize the DBN.

Finally, all analytic nodes of the AG are converted in non-temporal nodes in the DBN, each having the same outcomes as the temporal node to which they are linked. Section 5.1 describes how these nodes can be parameterized to capture the accuracy of the corresponding analytic. Analyzing the network topology, we observe that the dependencies between attack steps are modeled through temporal arcs, which capture the delayed effect between the completion of one technique and the initiation in the next one. In contrast, nodes representing analytics are connected to their corresponding attack step via immediate arcs, since the observable evidence of an attack step is assumed to be available instantaneously once the step is executed.

3.3.2 Modeling the Power System

There are two main goals of the second section of the DBN:

1. To model the state of the power grid components targeted by the attacker (e.g., switches and circuit breakers).

2. To represent how changes in state of these controlled devices are reflected on the power

system measures, thereby enabling a predictive model that can assess impact of the attacker's strategy over time.

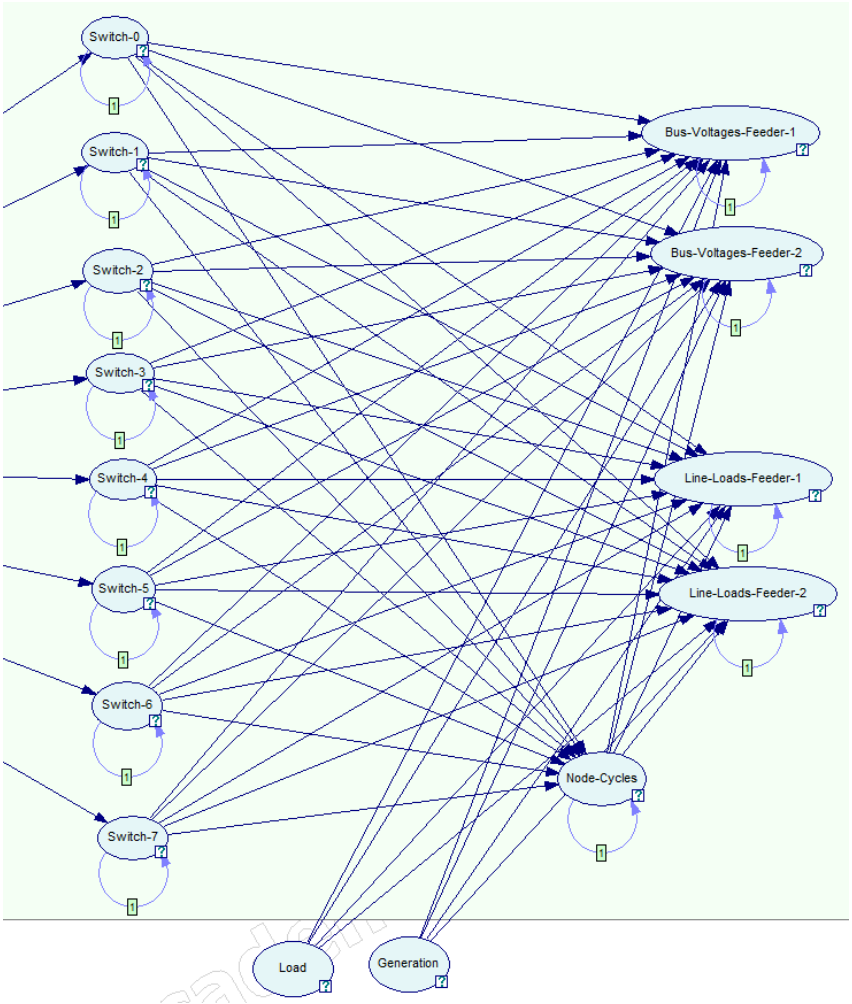


Figure 4: Second component of the DBN modeling Power System related aspects, including switch positions (*Switch-[i]*), the percentage of busses or lines operating out of specification for each Feeder (*Bus-Voltages-Feeder-[i]* and *Line-Loads-Feeder-[i]*), information about the presence of cycles in the power grid topology (*Node-Cycles*) and generation/load profiles (*Load* and *Generation*)

Following this approach we begin from analyzing the DBN nodes that model the switches in the power grid introduced in Section 3.1. These nodes are labeled as *Switch-[i]*, where the i index (ranging from 0 to 7) corresponds to one of the 8 switches present in the power model. Each of these temporal nodes (equipped with a self temporal loop to capture their evolution over time) has two possible outcomes representing the state of the corresponding switch: open (`O`) or closed (`C`). These nodes are directly influenced by the initiation of the *ManipulationOfControl* attack step, to which they are directly connected by immediate

arcs. Altering the the state of the switches in an uncontrolled manner directly impacts the power grid topology, potentially causing instabilities in bus voltages and line loads and ultimately leading to extensive blackouts. This behavior is modeled by different nodes in the DBN. A temporal node *Node-Cycles*, which is directly connected to each *Switch-[i]* node by immediate arcs, provide information about the percentage of busses that belong to at least one minimal cycle in the power grid topology. This is represented through four outcomes:

1. `NC_0`: from 0% to 25% of busses belongs to at least one minimal cycles.
2. `NC_1`: from 25% to 50% of busses belongs to at least one minimal cycles.
3. `NC_2`: from 50% to 75% of busses belongs to at least one minimal cycles.
4. `NC_3`: from 75% to 100% of busses belongs to at least one minimal cycles.

The *Node-Cycles* node is connected to four additional temporal nodes that represent the percentage of busses and lines operating outside specification, with a separate node defined for each feeder (see Section 3.1 for further details on the power grid structure). The nodes *Bus-Voltages-Feeder-1* and *Bus-Voltages-Feeder-2* model the percentage of buses whose voltage level falls outside certain limits. Each node has four outcomes:

1. `BV_0`: from 0% to 25% of busses out of specification.
2. `BV_1`: from 25% to 50% of busses out of specification.
3. `BV_2`: from 50% to 75% of busses out of specification.
4. `BV_3`: from 75% to 100% of busses out of specification.

Line-Loads-Feeder-1 and *Line-Loads-Feeder-2* model the percentage of lines whose load surpasses 100%, in each feeder. As before each of these nodes has 4 outcomes:

1. `LL_0`: from 0% to 25% of lines out of specification.
2. `LL_1`: from 25% to 50% of lines out of specification.
3. `LL_2`: from 50% to 75% of lines out of specification.
4. `LL_3`: from 75% to 100% of lines out of specification.

Finally two non temporal nodes, *Load* and *Generation*, respectively model the load and generation profile through 4 discrete outcomes each. We further clarify this concepts in Section 4.

3.4 Generalized Stochastic Petri Net model

The complete GSPN adopted for our experiments can be divided in three parts. The pieces describing attack and the post-attack phases, which can be derived directly from the AG described in Section 3.2, represent the core of the network and follow the path structure of the complete Industroyer attack, as explained in Section 3.4.1 and 3.4.3. The last part of the GSPN describes the power grid through a simple model focused on the state transitions caused by the attack, as described in Section 3.4.2. As it follows, this choice contributes to diversify analysis on GSPN from the one performed on DBN: GSPN model was designed

to offer a more targeted analysis on attack steps sequence and relations between steps and evidences.

However, one could vary the GSPN structure in order to **reduce or extend the net by *tag-based* composition**. This feature is useful for analysis focused on specific components of the network, which can be treated as independent pieces. For example one can include the pre- and post-attack phases,[2] only when needed, by simply combining independent networks representing the phases. Furthermore, the GSPN representing the impact of the attack on the power grid (corresponding to the *PowerSystemFailure* node of the attack graph) is also independent, so a more or less complex models can be adopted as needed, if the simple one described in Section 3.4.2 is not sufficient.

3.4.1 Modelling the attack phase

The basic structure of the GSPN is obtained directly from the AG described in Section 3.2. An example of such mapping for the first two attack steps is shown in Figure 5. Starting from an initial state in which the attack is assumed not to have begun yet, the left part of network represents the completion of the first attack step, called *UnsecCred* in the AG of Section 3.2, using the exponential transition (in white) *TUnsecureCred* to change the marking from a generic *Infected* place to the *UnsecCred* place which represents the step completion. The exponential transition *TUnsecureCred* firing causes a delay in the system which is characterized by the transition's rate parametrization. Next, the two couples of immediate transitions (in black) separating the state *UnsecCred* from *EndS1* model all the possibilities (with all information on the events probability) that the linked evidence, *FileAccessUC*, will trigger because of the step *UnsecCred* completion. Briefly, the first couple models the paths which depend on the evidence state (each evidence can be on or off), while the second models the evidence (probabilistic) trigger. The same logic is then repeated for the nex attack step, *ReadConfiguration*, and for each attack step in the Industroyer attack graph of Figure 2.

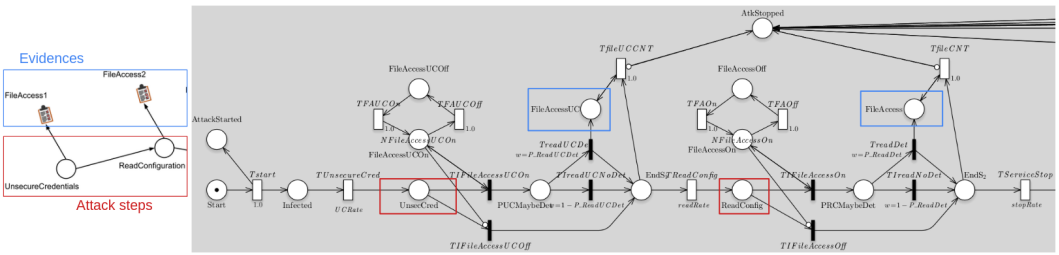


Figure 5: An example of mapping from the Industroyer AG (Figure 2) to the GSPN model for the first two steps of the attack.

Given the main structure of the GSPN derived from the AG, the complete model has been then expanded with additional elements in order to empower the possibilities for model checking and to emphasize the differences between DBN and GSPN models:

[2]Excluding the *UnsecureCredentials* steps assumed in Section 3.2, in this research we didn't explore preconditions and steps needed to correctly infect the system with Industroyer. However, one can easily extends AG, DBN, and GSPN as well, to include all the steps to be considered for the analysis by applying the same logic exposed for the attack and the post-attack phase.

- **Each evidence can be turned on or off** before or during the attack. The piece of net that models this aspect for the first evidence is represented in Figure 5 by the states *FileAccessUCOn* and *FileAccessUCOff*. By directly changing the initial (and parametric) marking of these two places before performing analysis, it is possible to exclude one or more evidences without modifying the structure of the model. There are two edges between *FileAccessUCOn* and the immediate transition *TFileAccessUCOn*, so the token is restored after the step completion. This feature could be helpful to declare some very specific query focused on the evidence state, such as "what's the probability that Industroyer caused a power grid crash if all the evidences remained turned on during the entire duration of the attack". However, if the analysis doesn't include such queries it is possible to use a different net where it was removed the edge that restore the token in *FileAccessUCOn* and it was added an edge from *FileAccessUCOff* to *TFileAccessUCOff* (thus, the inhibitor edge from *FileAccessUCOn* to *TFileAccessUCOff* become useless and it can be removed to have a cleaner model). In this case, token in the on/off structure is fully consumed by taking one of the possible paths and this significantly reduces the state numbers of the GSPN reachabilty graph. By applying this optimization on all the evidence on/off structures (as made in Figure 7) of the GSPN, the number of states in reachability graph drastically drops by a factor of five as shown in table 2 , which leads to an enhancement in model checker performance. Besides, similar queries to the one presented above can still be specified, but by using DTAs more complex than ones applied to the model which not consume tokens from on/off structure.

- **The evidence trigger can potentially stop the attack**. For each step S_x, the evidence associated with step S_{x-1} may lead the system to the interruption of the attack before the completion of x, reaching a final state called *AtkStopped*. In the model shown in Figure 5, the success of the attack depends on the parameters associated to the transitions *TfileUCCCNT*, and *TfileCNT* (which stop the attack) compared to parameters associated to *TReadConfig* and *TServiceStop* (which take the attack sequence to the next step). With this addition, the model supports questions about how much it is possible to stop the attack or recover the system, thus modeling for each step a **mutually exclusive competition relationship** between the progress of the attack and its mitigation. This is one of the main feature of the GSPN model compared to DBN which could be extended with a similar concept, but with a huge cost in overcomplicating the model by including an *attackStopped* state connected to all the evidence and attack steps.

- **Additional places for model checking purpose were added**. Excluding the place *AtkStopped*, which is very useful also in declaring expressive queries, two additional places (which are not directly part of the source AG) can be added to the GSPN to simplify DTA used in CSL^{TA} model checking. The idea behind these addition is to preserve information about certain "milestones" reached in the paths evaluation. Indeed, the two additional places added are called *AttackStarted* and *AttackDetected* and they are respectively meant to track that the attack is started and how many evidences were triggered (this is particular useful to express queries about the relationship between the defense system capabilities of finding out the attack and stopping the attack). *AttackStarted* is activated directly when the first transition of the model called *Tstart* fired, while each immediate transition meant for detection (such as *TreadUCDet* or *TreadDet*) fire implies a token addition to the place *AttackDetected*. Both

these two places have no impacts on the total number of markings.

- **Different *manipulation of control* strategies are included** in the model. Following the description of Industroyer behavior from [36] the GSPN portion representing the *ManipulationOfControl* step is composed of more than a single strategy, each of them directly derived by following the mapping from AG to GSPN explained above. As figure 6 shown, each strategy can be then choosen from the place *ReadyToAttack* according to the weights of the immediate transitions: in this way a single strategy can be adopted (by setting all the weights to zero except for the one associated to the relevant strategy), as in the real Industroyer malware attack where the strategy were included directly in the initial configuration file read in the second attack step of the AG, or the choice can be fully probabilistic. Besides, there are some additional assumptions compared to AG and DBN models, such as each strategy has its own on/off switch for monitoring mechanism, as in a real scenario where one could set up specific alarm on network traffic monitoring to identify suspicious packets sequences. Nevertheless, the evidence associated (the place called *NetworkAnalysisAlarm*) is the shared between all the strategies. Again, this modeling decision has been made to be much closer to a real scenario where each alarm can be on or off, but in general terms all of them can identify the same situation: a dangerous sequence of packets (and so, a single alarm trigger is sufficient to fully raise the evidence). Finally, at the end of the *ManipulationOfControl* step there is a cycle which come back at the end of *ServiceStop attack*, from which two different paths can be taken distinguished by the malware behavior: in one case Industroyer repeats the attack on power grid,[3] in the other it proceeds to post-attack phase. In the GSPN shown in figure 6, first and second strategies can be distinguished for the command given by all the IEC-104 packets (ON in the first scenario, OFF in the second one), while the third strategy consists of alternating ON packets and OFF packets.[4]

3.4.2 Modelling the power grid

As discussed before, the GSPN portion representing the power grid can be composed to the attack network. It is then conceptually independent and one can decide to change it to have an analysis more focused on the power grid characteristics. For instance, in the GSPN presented here the portion modeled to represent the power grid (the lower right part of Figure 6) is not so expressive as in the DBN. Indeed, it is modeled through three simple states: *NetWorking*, *NetAlert*, and *NetDanger*, whose evolution basically depends both on the steps of the attack that directly impact the power grid and on the power grid evolution itself. About this last kind of influence, it is assumed that the network could naturally change its states if there's a sufficient amount of time. This is especially interesting if some recovery mechanisms from a dangerous state are implemented at lower level than the control room one.

[3] According to the tokens initialization in the place called *counterNetAttack*, which works as a counter it could be helpful to analyze attack scenarios with strategy variations or prolonged attacks over time, but it is also useful considering the power grid model presented in Section 3.4.2 which cannot be necessarily set in a "danger" state in a single step iteration. In conclusion, the utility of *counterNetAttack* strictly depends on the power grid model adopted.

[4] For the sake of simplicity, the three strategy modeled are supposed to attack all the switches in the power grid with the same packets sequence. As described in [36], in reality Industroyer can be configured to attack by following a specific strategy for each switch and each substation. All the additional strategies that one wants to include can be added to the net by building the same structure schema of the ones presented.

	Tangible Markings	Vanishing Markings	**Total Markings**
GSPN without consuming on/off tokens	89472	42342	131904
GSPN consuming on/off tokens	18432	10752	29184

Table 2: A comparison between the total number of markings obtained by calculating the reachability graph of the GSPN models proposed. First model keeps tokens in all the on/off switch structure of the evidence, while the second model consume tokens from on/off structure as soon as the decision in path evaluation between evidence state is taken.

Focusing on the immediate transitions between the end of each *ManipulationOfControl* strategy and the power grid, here the transitions are immediate because the time needed to influence the net is already passed with the prior exponential transition firing (for example, TM_1 in the case of the first strategy). Each immediate transition conveys the probability, which depends on the specific strategy adopted, of influencing the power grid in a certain way:

- From the *working* state to the *alert* one. Power grid in an *alert* state is still working at all, but with some configurations not allowed or potentially dangerous, as in the case of having some unwanted, but still non-critical, cycles in the power grid topology. For example, the immediate transition representing this influence for the first strategy is *TM1extWtoA* in Figure 6.
- From the *alert* state to a *danger* one. Power grid in a *danger* state is considered not working and fully crashed. Reaching this state could be considered the Industroyer goal in the context of the model checking. The immediate transition representing this influence for the first strategy is *TM1extAtoD* in Figure 6.
- No change of states from the actual one of the power grid. In this case the *manipulationOfControl* was not sufficient to change the power grid state. The immediate transition representing this influence for the first strategy is *TM1doNothing* in Figure 6.

Actions are reduced at minimum in the example GSPN presented in order to put much effort in the conflicts between attack success and evidence effectiveness in stopping the attack, here lies the main reason because of we didn't model additional possibilities such as the direct change from *working* to *danger* or a nested sequence of immediate transitions to represent better the real probability of not influencing the network given the actual state of the power grid.

3.4.3 Modelling the post attack phase

The last portion of the GSPN (Figure 7) models the post-attack phase composed by the steps *CreateModSysProc* and *LossOfControl* from the AG in Figure 2. All the considerations made in Section 3.4.1 for the main attack phase are also valid here. In particular, the

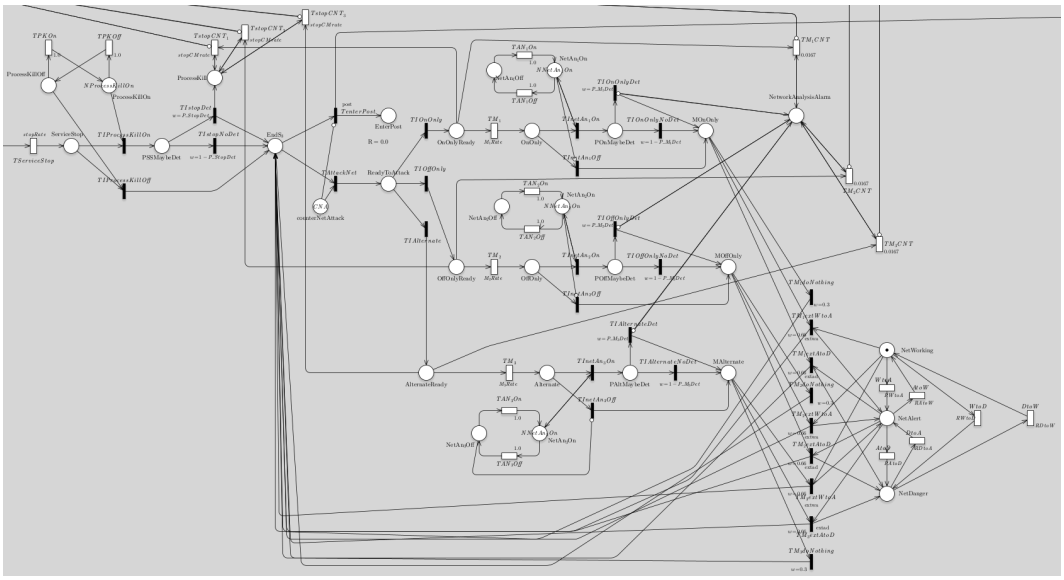


Figure 6: The portion of GSPN modeling the steps *Service Stop* and *ManipulationOfControl* from the Attack Graph (Figure 2). Each of the three possible strategies for the step *ManipulationOfControl* has a direct impact on the power grid, which is represented in the GSPN portion shown the lower right part of the image.

following additional places have been added to enhance model checking and allow the post-attack phase interruption by the evidences trigger:

- The place *EnterPost* keep the information of Industroyer phase change.
- The place *postAtkDetected* counts how many evidences were triggered. Exactly as explained in Section 3.4.1, token can be added to this place from each immediate transition firing that trigger an evidence.
- The place *Recovery* is the reciprocal of the *AttackStopped* in the Figure 5 and it respresents the post-attack interruption. This depends on the conflicts between the firing of the exponential transitions that leads from a step to its next one and the firing of the countermeasure (like *TLossCtrl* and *TmodSysCNT* in Figure 7). It is also important to notice that system recovery can be eventually made even after the end of the last attack step (*LossOfControl*) and before reaching the final state of the GSPN named *End*. The idea here is to assume that it is always possible to recover the system if one notice that the system is down and if there's sufficient amount of time to do so. The conflicts between transition *Tend* (parametrized with a variable rate named *endRate*) and timed transitions which leads to *Recovery* model this "last opportunity" to save the infected control room.

One unique feature of the last step in the post-attack phase compared to all other steps presented in the AG is the presence of two evidences. Here the mapping between AG and GSPN is conceptually the same as explained above, however it is necessary to consider all the combinations of evidence states, since one can have both the evidences on/off or

only one on (for a total of four basic cases). Then, for each of these paths, there's the probabilty of the evidence trigger which vary according to the path taken: a branch where nothing is triggered if both the evidences are off, a single trigger (or not, depending on the specific evidence accuracy) in the case a single evidence is on (two different branches for each evidence) and finally the last branch leads to the four possible cases when both the evidences are on.

Besides, observing how *WinRegKeyMod* can be marked in the GSPN, it is possible to notice that this place representing evidence is shared between the two attack steps *CreateModSysProc* and *LossOfControl* (there is only a single on/off structure which control the evidence state). Indeed, the main difference between the basic mapping (even if assuming two evidences on the same steps as for *LossOfControl*) from AG to GSPN presented in Section 3.4.1 and the one made for post-attack phase can be found in a specific interpretation of the evidences *WinRegKeyMod1* and *WinRegKeyMod2*. In this context, the evidences were unified in a single one which is named *WinRegKeyMod* in the GSPN. The reason behind this choice can be found in the nature of the evidence in a real scenario: a malicious setting of a register contents, if monitored and identified, could be immediately interpreted as a clear attempt to cause a total breakdown of the targeted machine, regardless the source of the malicious action and the specific register changed. Given that this effect is exactly the goal of the last step of the Industroyer behavior (*LossOfControl*), so it is possible to assume that active countermeasures for register modifications (even if triggered in the previous steps, *CreateModSysProc*, and possibly a different register than one or more changed for *LossOfControl* step) can allow a recovery after the control loss of the targeted machine. That's why the evidence is assumed to be shared completely between the two attack steps.

From the model perspective, this assumption has no impacts on the combinatorial structure included in the piece of GSPN model which control the *LossOfControl*'s evidences to model all the cases allowed with two evidences on a single step, but it is necessary to add additional inhibitor edges on immediate transitions to always ensure a bound of one token maximum on the shared evidence *WinRegKeyMod*.

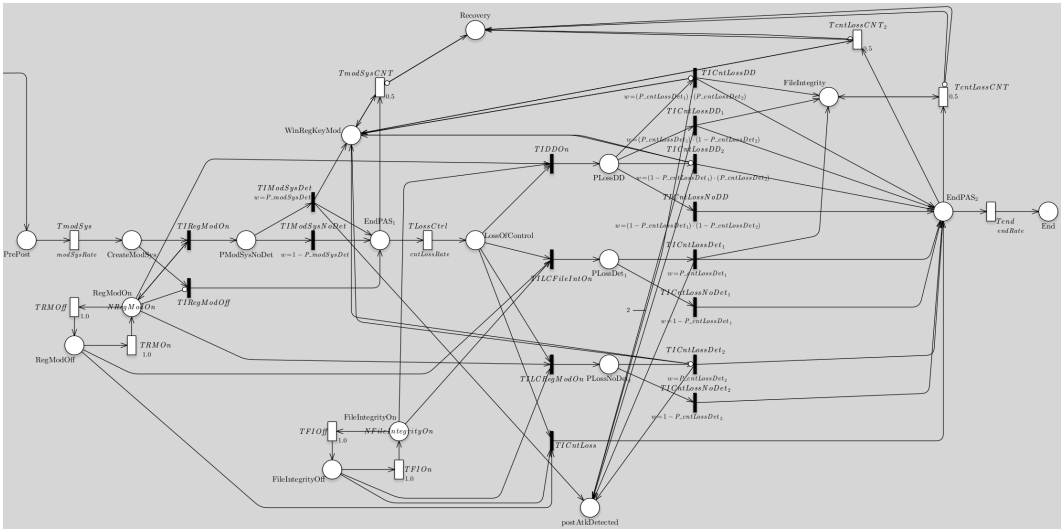


Figure 7: The portion of GSPN modeling the steps of the post attack phase shown in the Attack Graph (Figure 2).

4 Attack emulation and traces extraction

Gathering high-fidelity information describing both normal and under-attack conditions of the power grid is a crucial step to be able to train our models. The available datasets, such as [23, 26] cover a wired spectrum of power system related cyberattacks and do not focus specifically on reproducing all possible strategies that the Industroyer 1 malware could employ. In addition, they often lack exhaustive information on the load and generation profiles used, which are essential to understand the baseline operational condition of the electrical grid. For these reasons we choose to develop an extension of the Wattson framework capable of generating synthetic traces of multiple variants of both normal and attack scenarios, which were then used to train our inference models.

4.1 Emulating Industroyer in Wattson: wautorunner

The main idea behind wautorunner is to develop a Wattson extension that allows the emulation of different Industroyer 1 attack strategies in different power system conditions. Figure 8 illustrates a scheme of the wautorunner workflow. We start from a template scenario (shown in the left part of the figure), which in our case is the *CIGRE MV all DER* model that was already presented in Section 3.1.

Wautorunner takes as input the configuration files describing the power grid (including initial setup and applied power profiles), the ICT network topology and initial script controller configuration (a Wattson component enriched to emulate the Industroyer 1 cyberattack). This base scenario is then altered by the wautorunner module, applying different sets of *modifiers*, defined using the following strategy and that alter multiple aspect of the starting configuration:

- `ExecTimeModifier(80s)`: Defines the total duration of the emulation run, after which Wattson is sopped.

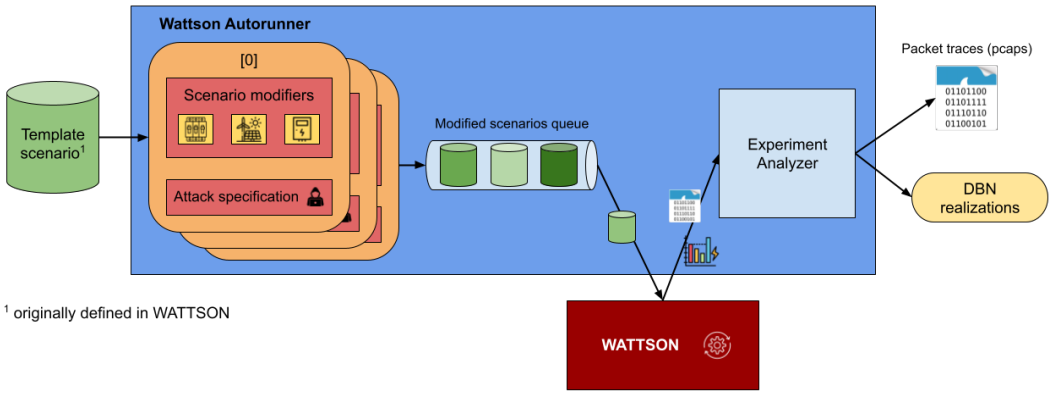


Figure 8: Scheme of the wautorunner module workflow

- `SimulationIntervalModifier(1s)`: Sets the time interval between consecutive power simulation. This corresponds to the discretization step discussed in Section 5.1.
- `SetMinMaxVoltageModifier(0.95, 1.05)`: This modifier resets the voltage limits of each bus between 0.95 p.u and 1.05 p.u. These are commonly used thresholds in literature and are used to define the "out-of-specification" voltage conditions for each bus.
- `MultiplyMaxCurrentModifier(0.5)`: Reduces the maximum current limits of each line to 50% of its original value. This just a way of configuring more sensitive overload thresholds for each line.
- `SetSwitchesModifier(switchConfig)`: This modifier sets the initial stable switches configuration. In our case switches 1, 2 and 4 are open and the others are closed.
- `MultiplyLoadsModifier(uniform(0.75, 1.75))`: It sets the scaling factor of each load in the power grid, drawing it from a uniform distribution bounded in $[0.75, 1.75]$. By exploiting this strategy, in each run, we can emulate different load conditions under which the power system operates.
- `MultiplyGenerationModifier(uniform(0.75, 1.75))`: It sets scaling factor of each generator in the power grid, extracting it from a uniform distribution bounded in $[0.75, 1.75]$. By exploiting this strategy we can emulate different generation conditions under which the power system operates.

Apart from the initial configuration, a specific modifier called `AttackerStrategyModifier` is responsible for choosing and configuring the type of strategy followed by the emulated Industroyer malware for the current execution. The attack strategy defines the type of IEC-104 command messages sent by the attacker and the time interval between consecutive messages. Every type of available strategy selects a certain number of switches to be attacked, drawing it from a uniformly distributed discrete random variable ranging in $[2, 8]$ and then a specific set of target switches is randomly identified. Finally one of the following strategies is randomly chosen:

1. `N` (No action): No commands are sent so the power grid continues to function following its initial configuration.

2. `Alternate`: The malware repeatedly modifies the state of every target switch with a time interval t drawn from a uniform continuous distribution in $[2s, 5s]$ (coherently with the values reported by the analysis we referred to) and with a random start delay bounded in $5s, execTime - 5s$.

3. `OpenOnly`: This strategy is similar to the `Alternate` one but only command opening the target switches are sent to the RTUs.

4. `CloseOnly`: Also this strategy is similar to the `Alternate` one but only command closing the target switches are sent to the RTUs.

5. `Explicit`: This strategy was not directly related to the Industroyer 1 malware, but we added it to simulate an attack more similar to Industroyer 2. In this situation, for each target switch, a point in time is drawn following a uniformly distributed random variable in $5s, execTime - 5s$ and just a single command will be issued to alter its original state. This is a less detectable and more protocol-compliant attack strategy.

Notice how each one of these previously mentioned attack strategy corresponds to an outcome of the `ManipulationOfControl` node we briefly mentioned in Section 3.3.

After the original template scenario has been modified by applying the aforementioned modifiers, it is provided to Wattson that runs it according to the configuration. At the end of each run, an *Experiment Analyzer* module is responsible for collecting all artifact generated by the emulation, including packet captures and power measurements that will be used to generate traces to parameterize the DBN model.

4.2 Traces generation for DBN parameterization

Generating discrete traces for DBN parameterization is the last task of the emulation environment. As we will see in Section 5.1, discrete traces are required only for the power grid related section of the DBN. This means that, starting from the simulated power grid measurements falling each time slice, the corresponding outcome for each node must be determined. For example, with a total execution time of 80 seconds and a discretization time step is 1 second, there will be 80 time slices. During trace generation, all measurements generated by the emulator are assigned to their respective time slices; if multiple values for the same metric occur in a slice, the last recorded value is retained. At this point each outcome is computed based on its semantic. For example each *Switch-[i]* node, its outcome `O` or `C` will reflect the state of the i^{-th} switch in that time slice. The outcome of the node *Node-Cycles* is determined by the percentage of busses belonging to at least one minimal cycle. Also the outcome for the nodes *Bus-Voltages-Feeder-[i]* and *Line-Loads-Feeder-[i]* will be computed based on the percentage of out-of-specification bussed and overloaded lines, respectively, in each feeder. Finally, it is worth mentioning how the single outcome for the non-temporal nodes *Load* and *Generation* is computed. We saw in Section 4.1 that a load and generation factor is applied to each execution run. Those factors are used to establish the outcome of the corresponding node depending on the interval it belongs to: `L_0/G_0` $\rightarrow [0.75, 1)$, `L_1/G_1` $\rightarrow [1, 1.25)$, `L_2/G_2` $\rightarrow [1.25, 1.5)$ and `L_3/G_3` $\rightarrow [1.5, 1.75]$.

5 Models parameterization

The final step before performing inference is the parameterization phase. This section describes the methodologies used to estimate the parameters of the models, based on the assumptions made while modeling the attack steps and the discrete traces generated by the wautorunner module.

5.1 DBN Parameterization

Setting the a priori probabilities for DBN nodes with no ancestors and the conditional probabilities of all the other nodes is a crucial step to ensure trustworthy inference results. The CPTs of all the attack step derived nodes are defined in a similar way as the state of a node depends on both the state of its parents and on its own state at the previous time slice. We call p_s the probability that node s is in a successfully completed or initiated attack state. For nodes with binary outcomes, this corresponds to the probability of the `C` outcome. For multi-strategy nodes (such as *ManipulationOfControl*), p_s refers to the cumulative probability of being in one of the outcomes associated with a specific strategy (`Alternate`, `OpenOnly`, `CloseOnly`, or `Explicit`). For each node we also define if it is modeled as persistent or non-persistent. A node is persistent if it stays completed / started even if its preconditions stop being valid, otherwise it is modeled as not persistent. We define p_p as the probability of staying persistent and we set it to 1 for persistent nodes or 0 for non-persistent nodes. This probabilities depends on the specific attack step. Let's consider, for instance, the CPT of *ReadConfiguration* node in Table 3. At time t (ulterior layer), it depends on itself and *UnsecureCredentials* at time $t-1$ (anterior layer). Each row of the CPT represents a specific state configuration of *UnsecureCredentials* and *ReadConfiguration* at time $t-1$ and provides the probability that *ReadConfiguration* evolves towards the state `U` or `C` at time t. In rows 1-2, *UnsecureCredentials* has not happened yet, so the probability that *ReadConfiguration* occurs (state `C`) at time t is null. In rows 3-4 *ReadConfiguration* has occurred but *UnsecureCredentials* it not active anymore at time $t-1$ so the probability that *ReadConfiguration* stays active at time t depends on the probability of persistence of the node *ReadConfiguration* ($p_{p(RC)}$). In rows 5-6 *UnsecureCredentials* has occurred and *ReadConfiguration* is not active at time $t-1$; so *ReadConfiguration* may occurr (`C`) or not (`U`) at time t with probability p_{RC}, i.e. p_s with s =*ReadConfiguration*. In rows 7-8, *ReadConfiguration* and its precondition *UnsecureCredentials* are already active at time $t-1$, so it will maintain its state.

Recall that analytics are modeled with non temporal nodes and that occurrences of attack steps raise the alarm of connected analytics. Analytics are imperfect: to each technique an accuracy value (*acc*) in $[0, 1]$ models the probability of that analytic being correct (true positives and true negatives), while the probability of the attack going undetected or of the analytic raising even when the corresponding attack step did not occur is $1 - acc$. The resulting CPT of a generic analytic node is structured as in Table 4.

We will not explicitly detail the case of the multi strategy node *ManipulationOfControl*; however in this situation the probability of successful initiation of a strategy (p_{MoC}) is equally distributed across all the strategy-related outcomes (`Alternate`, `OpenOnly`, `CloseOnly`, or `Explicit`). The probability of persistence ($p_{p(MoC)}$) corresponds to the likelihood of remaining in the currently initiated strategy after its precondition is no longer valid. We assume that, once a strategy has been started, it is not possible to switch to another strategy except from aborting it. Finally the analytic *NetworkTrafficAnalysis*, associated to the

	time $t-1$		time t	
	UnsecureCredentials	*ReadConfiguration*	*ReadConfiguration*	pr.
1	U	U	U	1
2	U	U	C	0
3	U	C	U	$1 - p_{p(RC)}$
4	U	C	C	$p_{p(RC)}$
5	C	U	U	$1 - p_{RC}$
6	C	U	C	p_{RC}
7	C	C	U	0
8	C	C	C	1

Table 3: CPT of *ReadConfiguration* node.

	Technique	**Analytic**	**pr.**
1	U	U	acc
2	U	C	$1 - acc$
3	C	U	$1 - acc$
4	C	C	acc

Table 4: CPT of a generic analytic.

node *ManipulationOfControl*, has its probability of raising a false alert equally distributed across all attack strategies.

Using the p_s value to parameterize each attack-step node of the DBN simplifies the parameterization process, but it raises the challenge of deriving, learning or providing a realistic estimate of p_s. Obtaining such estimate from domain experts or from measurements on real systems is often a quite difficult and error prone task. A key complication is that DBNs are discrete-time models, whereas attack processes are more naturally modeled in continuous time. For this reason to derive the conditional probabilities we resort to an ideal *continuous* time model, which we approximate with the proposed DBN in a discrete time domain. Accordingly, we compute p_s based on an estimate of the conditional mean time of completion / initiation of each step, i.e. of the time of completion dependent on the satisfaction of attack preconditions. The approximation ensures that the mean completion / initiation time of the attack-steps computed in the resulting discrete model match the mean values provided as input parameters. This continuous-to-discrete approximation is based on the *uniformization method* [29] which replaces an exponentially distributed random variable in continuous time using a geometrically distributed variable in discrete time, where each discrete step has a fixed duration. When the time step is sufficiently small, the geometric distribution closely approximates the exponential distribution, which is completely characterized by its rate (or equivalently its mean time). We also model the execution of multiple parallel attack steps where just the shorter one is accomplished, even though Industroyer 1 primarily presents sequential activities. Given a collection of mean completion times, the continuous time approximation is derived using:

$$\Delta t = \frac{1}{m \sum_i (1/\overline{T}_s)}; \qquad p_s = \frac{\Delta t}{\overline{T}_s} \tag{1}$$

Here Δt denotes the computed step-size of the approximating model, $\overline{T}_s$ is the desired mean completion time of attack-step s in the continuous model, and p_s is the corresponding attack probability in the DBN. The parameter m, set to 1 for our experiments, controls the accuracy of the approximation. Further technical details on this approximation can be found in [7]. In practice, deriving mean completion times from domain expert opinions is a relatively straightforward process.

The final portion of the DBN to be parameterized is the power grid-related section. As discussed in Section 4, the discrete traces generated by the wautorunner module are used to learn CPTs of the nodes modeling the power system's behavior under both normal operations and attack conditions. This learning process is implemented by applying the Expectation Maximization (EM) algorithm, which iteratively estimates the theoretical probability distribution (i.e., the CPTs of the DBN nodes) that best fits the observed data. The EN algorithm alternates between two phases: an Expectation step (E-step) and a Maximization step (M-step). In the E-step probabilities of unobserved (missing) variables are inferred given the observed data and the current parameters. In the M-step parameters are updated by minimizing the Negative Log-Likelihood (NLL) function based on the fully observed data estimated during the E-step. The output of the algorithm is the final value of NLL, which measures how well the output distribution fits the sample data after reaching convergence. Since the EM algorithm is well established, we do not discuss it in detail here; further information can be found in [32]. The results of the parameterization are presented in Section 6.

5.2 GSPN Parameterization

As mentioned in section 3.4 transitions in GSPN must be parameterized according to their type. In particular, exponential (or timed) transitions are a key component of our model because of their impact on the conflict between the paths in which the attack fails or succeeds, respectively. All transitions representing the completion of an attack step, such as *TUnsecureCred* and *TReadConfig* in figure 5, are parameterized based on the **assumed average completion time** needed to complete the step. Specifically, the reciprocal of this time is calculated to obtain the rate to be associated with these exponential transitions. In GreatSPN, these rates can be specified using variables, such as *UCRate* and *readRate* associated to the aforementioned transitions taken as example. A similar argument can be made for all transitions leading to the *attackStopped* place: the average completion time to be considered to derive the rate is the one required for the system to be secured immediately after the associated evidence is triggered. This is a time that depends heavily on the countermeasures adopted at the system level to respond to certain malicious actions, such as the various steps of the Industroyer attack. Countermeasures are not a concept modeled within the AG in section 3.2, so no particular assumptions have been made about the specific way to stop the attack. Some general assumptions can then be done to choose these times: in real scenario, an alarm could trigger automatic actions to react the source of the warning communication, such as a block of user credentials that triggered the malicious action. In similar cases, time required to stop the attack can be really fast, which directly leads to a higher rate. In some other occasions, there could be processes where alarm triggers a warning to a human operator which has to analyze the situation and react accordingly: a significantly smaller rate than almost all automatic countermeasures. In the end, the way to choose these times depends heavily on the assumptions made for the scenario to be

analyzed. Assumptions on the mean time to complete a recovery process must be made also for the exponential transitions of the GSPN portion which represents the power grid (figure 6) and for all the evidence on/off switches described in section 3.4.1.[5]

While timed transitions parameterization has a direct impact on the time passage in the GSPN model, all the probabilistic information lives in the immediate transitions. However, not all transitions should be necessary parametrized. This includes all transitions that initiate mutually exclusive paths, as in all cases where it is evaluated whether an evidence is on or off (for example, *TIFileAccessUCOn*, *TIFileAccessUCoff*, *TIFileAccessOn* and *TIFileAccessOff* in figure 5) because the choice is completely deterministic based on the state of the places connected to the transitions. From there, the model can enter in a path where the evidence is on and then another choice must be taken between the evidence trigger or not (as for *TreadUCDet* and *TIreadUCNoDet* in figure 5). Unlike the previous example, these transitions must be assigned a value that represents the accuracy of the evidence, i.e., how likely it is that the evidence will trigger given the event represented by the step just completed. The evidence accuracy can be set in the GSPN modeled with GreatSPN through a dedicated variable associated to the transition that represents the detection (such as *P_ReadUCDet*). Then, the reciprocal transition representing the failure of the evidence to trigger is simply one minus this variable. For post-attack last step (figure 7), where two evidences are associated to the same steps, immediate transitions must be parameterized with the combination of each variable associated to the evidence, assuming a complete probabilistic independence between evidence trigger. An example could be seen in the right part of figure 7 where the multiplication between *P_cntLossDet1* and *P_cntLossDet2* establishes the probability of both evidences trigger after *LossOfControl* step.

The last set of components to be parametrized is the one of the immediate transitions which connect the GSPN attack phase to the power grid (as *TM1doNothing*, *TM1extWtoA* and *TM1extAtoD* for the first strategy of the step *ManipulationOfControl* in figure 6). Transitions here should be valued with a probability which represents the effectiveness of the attack (consisting of all strategies considered during a massive sending of IEC-104 packets to the substations of the power grid) in changing the power grid state between three options labeled as *working*, *alarm* and *danger* as described in section 3.4.2. Considering a power grid model such as the CIGRE MV (figure 1) we used in Wattson simulation, there are many switches configurations which can be classified as dangerous or alarming. Therefore, theoretically, it would be possible to extract the probability of reaching these particular configurations by simulating a certain attack strategy applied to different initial configurations of the electrical network. We are still exploring an extension of wautorunner (section 4.1) to parametrize the immediate transition based on the traces of the attack emulation. Despite this, the model can already be used to obtain interesting results by assuming the probability of impact of each strategy included in the model to parameterize these transitions. More precise details on GSPN initial markings and weight values chosen for the model checking will be provided in section 6.2.

[5]here the time is the one required to turn on or turn off an evidence. Some evidences can be activated almost instantly, while others could require longer time as in the case of awaiting approval for the evidence deactivation or the propagation time needed to update multiple servers with the configuration.

6 Results

This Section presents the inference results obtained from both parameterized DBN and GSPN models. We evaluate multiple scenarios and address different queries, exploring the differences and peculiarities of both models.

6.1 DBN Inference Results

Section 5.1 explored the parameterization techniques applied for the DBN model, but it did not specify the exact parameters adopted in our experiments. Table 5 summarizes the Time to Completion/Initiation (TTC/I) values for each attack-step–related node, together with their associated persistence condition. We recall that a node TTC/I defines the expected time required for the corresponding attack step to succeed once its preconditions are met, while the persistence condition specifies whether the node stays completed / initiated even when its preconditions are no longer satisfied.

Technique/DBN Node	**TTC/I (s)**	**Persistence**
UnsecureCredentials	15	/
ReadConfiguration	10	✓
ServiceStop	10	✓
CreateModSysProc	12	✓
LossOfControl	25	✓
ManipulationOfControl	14	✗

Table 5: TTC/I (in seconds) and persistence condition for each attack step related node of the DBN, expressed using the following notation: ✓- persistent; ✗- non persistent; / - partially persistent.

While most techniques are modeled as persistent in their corresponding node of the DBN (*ReadConfiguration*, *ServiceStop*, *CreateModSysProc* and *LossOfControl*) the only node implemented as non persistent is *ManipulationOfControl*. This is because we assume it requires continuous SSH to be successfully executed, as the credentials obtained through the UnsecureCredentials step must remain valid and not be reset. The only exception is the node *UnsecureCredentials* which, despite having no preconditions to be satisfied, is considered partially persistent. We introduce a 1% likelihood that it reverts to the uncompleted (`U`) state, representing the possibility of the SSH credentials being reset.

In the following part of this section we will analyze the results obtained in different conditions and for different evidence streams provided.

6.1.1 No evidence

In this first scenario, we analyze the inference of the DBN model when no evidence is provided, so the model just relies on its prior information derived from the parameterization process. Figure 9 reports the posterior probabilities for different nodes in the DBN. In Figure 9a we show the attack success probability (likelihood of outcome `C`) for the attack step related nodes. These success probabilities increase over time accordingly to each node's mean TTC/I and its interdependencies. After almost 40 seconds, the success probability

of nodes *UnsecureCredentials*, *ReadConfiguration* and *ServiceStop* surpass 0.7. It is worth mentioning that the success probability of *UnsecureCredentials* settles just below 0.9 instead of reaching 1 due to its partial persistence discussed at the beginning of Subsection 6.1. For the post exploitation nodes, *CreateModSysProc* and *LossOfControl*, the model shows that they cannot be fully completed within the time window as their success probability just slightly surpasses 0.8 and 0.55 by the end of the inference, respectively. Analyzing the posterior probabilities for the outcomes of the *ManipulationOfControl* node in Figure 9b, it is worth noting that already after 40 seconds the cumulative likelihood having started any attack strategy (`OpenOnly`, `CloseOnly`, `Explicit` or `Alternate`) exceeds the probability of not having started it (`N`), but all attack strategies are equiprobable. Looking at the posterior probabilities of the outcomes of nodes *Bus_Voltages_Feeder_1* and *Bus_Voltages_Feeder_2*, in Figure 9c, we can clearly see that the attack has minimal impact on the percentage of busses out of specification. While the cumulative probability of having more than 25% of abnormal busses ($P(BV\_1) + P(BV\_2) + P(BV\_3)$) slightly raises towards the end of the inference for both Feeders, the single most likely outcome remains `BV_0` (from 0% to 25% of busses out of specification). A similar phenomena happens for the outcomes of nodes *Line_Loads_Feeder_1* and *Line_Loads_Feeder_2* (Figure 9d) modeling the percentage of overloaded or fully disconnected lines. In both cases the probability of outcome `LL_0` (from 0% to 25% of overloaded lines) remains the most likely outcome, even though for *Line_Loads_Feeder_1* the probability of outcome `LL_3` raises to be the second highest by the end of the inference. The limited impact of the attack is due to two main reasons:

1. uncertainty about which strategy was applied during the *ManipulationOfControl* attack step.

2. the missing evidence about load and generation profiles in which the power grid was operating. In situations of low load / generation the impact of the attack is reduced and the observable measured effects are smoothen-out.

Finally, analyzing the posterior probability of the *Node_Cycles* node (Figure 9e), we observe that after the *ManipulationOfControl* has started, the likelihood of having more than 25% of busses in at least a minimal cycle increases ($P(NC\_1)+P(NC\_2)+P(NC\_3)$), even though `NC_0` remains the most likely state. This is caused by the increase in probability of switch related nodes changing their state and introducing multiple cycles between busses in the power grid (their posterior probabilities are not reported for space reasons).

6.1.2 ManipulationofControl - OpenOnly

In this scenario we explore the situation where some evidence is provided that the *ManipulationOfControl* attack step has been successfully started performing the `OpenOnly` attack strategy. Table 6 reports the analytics raised during the inference, their outcomes and their time interval for which each outcome is valid. In this scenario the analytic *NetworkTrafficAnalysis*, associated to the *ManipulationOfControl* attack step, is implemented and highlighting no alarm (outcome `N`) from 0 up to 34 seconds, while it signals the presence of an `OpenOnly` attack (outcome `OpenOnly`) from time 35 to 79. For this scenario and all the others that will follow, some evidence will be set for the *Load* and *Generation* nodes on their outcome `L_2` and `G_3`, respectively. This last evidences provide the DBN some information that we want to infer the attack impact on a power grid which is already in medium and high generation and load conditions respectively.

Analytic	Outcome	Time range (s)
NetworkTrafficAnalysis	N	$0 \rightarrow 34$
NetworkTrafficAnalysis	OpenOnly	$35 \rightarrow 79$
Load	L_2	/
Generation	G_3	/

Table 6: Implemented analytics for the "ManipulationofControl - OpenOnly" scenario, including their outcome and time range (when aplicable).

Figure 10 shows the results produced by the current scenario. Figure 10a highlights that when the *NetworkTrafficAnalysis* analytic is raises at time 35, the success probability of the preconditions of the *ManipulationOfControl* attack step (*UnsecureCredentials*, *ReadConfiguration* and *ServiceStop*) spike to 1. Also the success probability of the post exploitation attack steps *CreateModSysProc* and *LossOfControl* increases reaching 1 and 0.7, respectively. This is caused by the fact that the initiation event of the *ManipulationOfControl* attack step has been slightly anticipated by the evidence provided, so the subsequent attack steps have more time to be completed. Figure 10b shows the probability of outcome OpenOnly reaching 1 at time 35 while the probability of outcome N (no attack strategy started) drops to 0. In Figure 10c the most likely outcome, for node *Bus_Voltages_Feeder_1*, appears to be BV_3 (more than 75% of busses out of specification in Feeder 1) reaching almost probability 0.7, while the other outcomes are equiprobable by the end of the inference. For node *Bus_Voltages_Feeder_2* the most likely outcome remains BV_0 (less than 25% of out of specification busses in Feeder 2) but after the *ManipulationOfControl* attack starts the probability of outcomes BV_1, BV_2 and BV_3 slightly raises cumulatively reaching 0.3. Figure 10d shows the probability for the outcomes of nodes *Line_Loads_Feeder_1* and *Line_Loads_Feeder_2*. For the first node the most likely outcome is LL_3 (more than 75% of overloaded or fully disconnected lines in Feeder 1) by the end of the inference. For the second node it is clear that the OpenOnly strategy lower the load on the lines of Feeder 2 as the most likely outcome becomes LL_0 just after the *ManipulationOfControl* attack starts, surpassing LL_1 (between 25% and 50% of overloaded or fully disconnected lines for Feeder 2), which was initially the most likely state. Finally Figure 10e shows that for node *Node_Cycles* the most likely outcome remains NC_0 (less than 25% of busses belonging to at least one minimal cycle) as the strategy of opening most switches reduces the likelihood of having cycles between busses.

6.1.3 ReadConfiguration - Failed

This specific case-study models the situation where the *ReadConfiguration* attack step fails in a specific point during the analysis. In particular, as it is reported in Table 7, its associated analytic node *FileAccess2* does not provide any information if the attack has been completed or not up to 29s. From 30s up to the end of the inference, some evidence that the *ReadConfiguration* step has not been completed is provided (outcome U). Non temporal evidence on *Load* and *Generation* nodes is set as in the previous experiment to L_2 and G_3 respectively.

Figure 11 shows the results for this scenario. In Figure 11a the success probability of multiple attack step-related nodes is shown. Before the evidence on *FileAccess2* is provided

Analytic	Outcome	Time range (s)
FileAccess2	U	$30 \rightarrow 79$
Load	L_2	/
Generation	G_3	/

Table 7: Implemented analytics for the "ReadConfiguration - Failed" scenario, including their outcome and time range (when aplicable).

at 30s, the success probabilities of all attack step evolve as in the "No evidence" scenario, but afterwards some interesting events can be highlighted. The success probabilities of the *ReadConfiguration* node and all subsequent attack steps (*ServiceStop*, *CreateModSysProc* and *LossOfControl*) drop to 0 as they did not have enough time to be completed before the negative evidence is provided. The success probability of the only attack precondition *UnsecureCredentials* instead stays at around 0.55 as the model considers the backward influence that the failure of the *ReadConfiguration* attack step could have been caused by the *UnsecureCredentials* attack step not being completed. Even in Figure 11b the probability of not having started an attack strategy of the *ManipulationOfControl* attack (outcome N) initially drops and then jumps to 1 as the evidence provides an indication that a precondition has failed. Figure 11c shows how almost all busses operate nominally in both Feeders as the most likely outcome is BV_0 in both cases. Just for Feeder 1 there is a probability of 0.4 of having between 50% and 75% of out of specification busses (outcome BV_2) but that's caused by the intense load and generation conditions. Figure 11d shows that, after the *FileAccess2* evidence is provided, in node *Line_Loads_Feeder_1* the only possible outcome is LL_0 (less than 25% of overloaded or fully disconnected lines in Feeder 1), while for node *Line_Loads_Feeder_2* the most likely outcome is LL_1 (from 25% to 50% of abnormal lines). This is caused by the intense load and generation condition specified as evidence and by the reduced number of lines of Feeder 2. Finally Figure 11e shows that for node *Node_Cycles* the only possible outcome after the evidence is provided is NC_0 (less than 25% of buses in at least one minimal cycle) as the power grid topology maintains its original configuration.

6.1.4 UnsecureCredentials - Reset

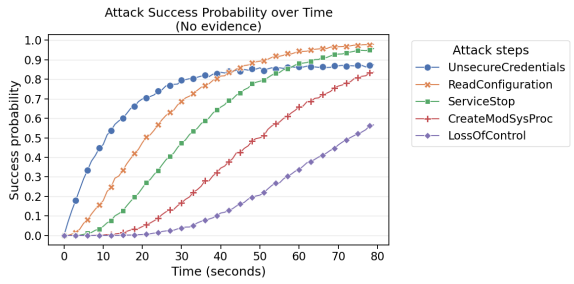
This last scenario models the situation where the SSH credentials, that were stolen by the attacker during the *UnsecureCredentials* attack step, are reset and his active SSH connection drops, possibly blocking all non-persistent attack steps depending from this precondition. As it is reported in Table 8, the *FileAccess1* analytic, associated to the *UnsecureCredentials* attack step, signals the absence of its corresponding attack step (outcome U) from 0s up to 24s. Then from 25s to 55s an alarm is raised (outcome C) from the same analytic. Finally the SSH credentials are rotated so from 56s the *FileAccess1* analytic is set back to state U. Non temporal evidence on *Load* and *Generation* nodes is set as in the previous experiment to L_2 and G_3 respectively.

Figure 12 reports the results of this last scenario. Figure 12a reports the success probabilities of multiple attack related nodes. It is clear that the *UnsecureCredentials* success probability closely follows the evidence provided, starting at 0, to then spike to 1 when positive evidence is provided, to finally drop back to 0 when SSH credentials get reset. It is interesting to analyze how the success probabilities of the *ReadConfiguration* and *ServiceStop* attack steps start raising accorting to their mean TTC/I just after the *FileAccess1* analytics

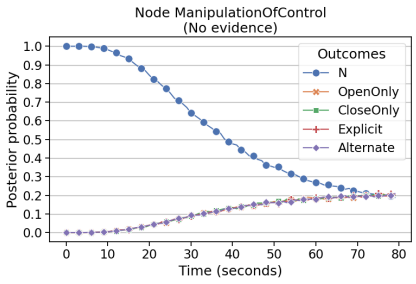
Analytic	Outcome	Time range (s)
FileAccess1	U	$0 \rightarrow 24$
FileAccess1	C	$25 \rightarrow 55$
FileAccess1	U	$56 \rightarrow 79$
Load	L_2	/
Generation	G_3	/

Table 8: Implemented analytics for the "UnsecureCredentials - Reset" scenario, including their outcome and time range (when aplicable).

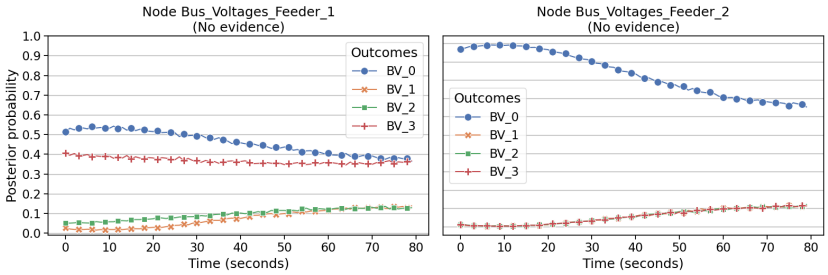
raises an alarm, but they do not drop after the analytic is reset with negative evidence. This is due to the persistent nature of both *ReadConfiguration* and *ServiceStop*, which was taken into account in the parameterization process as explained in Section 5.1. This means that after each of these nodes reaches its completed (C) state, it maintains that outcome even if its preconditions revert to the uncompleted (U) state. We recall instead that, the *ManipulationOfControl* node starting precondition is formed by the combination of the *ServiceStop* and *UnsecureCredentials* nodes through the *NodeAND* logical node. As both preconditions of the *ManipulationOfControl* attack step become valid (around 30s), the probability of one of its attack strategies being started raises. Ultimately the non-persistent nature of the *ManipulationOfControl* attack step results in a failed attack when the *UnsecureCredentials* node reverts to its uncompleted state, as it can be observed in Figure 12b. Instead, Figure 12c reports the probabilities of the outcomes for nodes *Bus_Voltages_Feeder_1* and *Bus_Voltages_Feeder_2*. While the posterior probabilities of BV_1, BV_2 and BV_3 start raising after the *ManipulationOfControl* attack step is started, they settle (for Feeder 1) or slightly decrease (for Feeder 2) after the attack is stopped due to SSH credentials reset (at 56s). By the end of the inference the most likely outcome is BV_0 (less than 25% of busses out-of-specification) for both Feeders. Figure 12d shows that, even for the *Line_Loads_Feeder_1* node, while the probabilities of its outcomes LL_1, LL_2 and LL_3 increase after the *ManipulationOfControl* attack step is attempted (35s), they settle or slightly decrease after the SSH credentials get reset. For the *Line_Loads_Feeder_2* node the probability of outcome LL_1 decreases and the likelihhod of outcomes LL_2 and LL_3 increases as the *ManipulationOfControl* attack is started, to then settle (for LL_1) or slightly decrease (for LL_2 and LL_3) after the SSH credentials are reset. Finally Figure 12e reports that the probability of outcomes NC_1, NC_2 and NC_3 increases after the *ManipulationOfControl* attack step is performed, but then settle at relatively low values when SSH credentials are reset. By the end of the inference the outcome NC_0 (less than 25% of busses in at least one minimal cycle) has still the highest likelihood (just below 0.8).



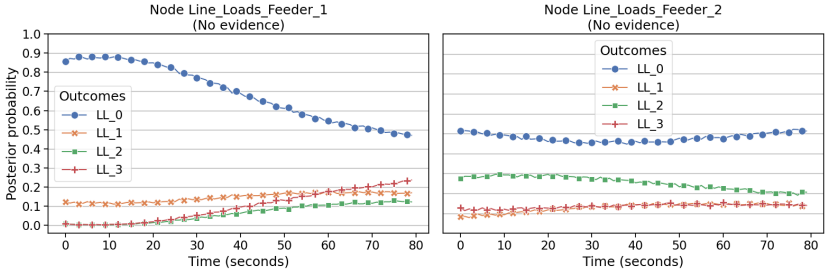
(a) Probability over time of outcome `C` for multiple attack step-related nodes.



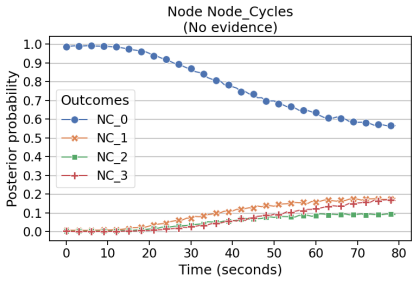
(b) Probability over time of the outcomes for node *ManipulationOfControl*.



(c) Probability over time of the outcomes for nodes *Bus_Voltages_Feeder_1* and *Bus_Voltages_Feeder_2*.

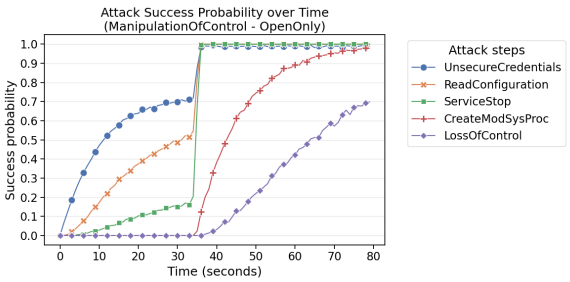


(d) Probability over time of the outcomes for nodes *Line_Loads_Feeder_1* and *Line_Loads_Feeder_2*.

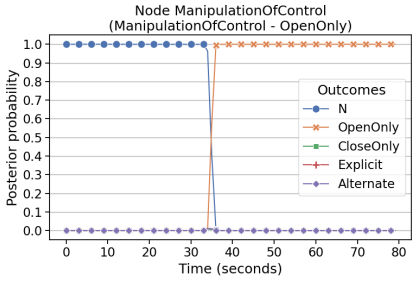


(e) Probability over time of the outcomes for node *Node_Cycles*.

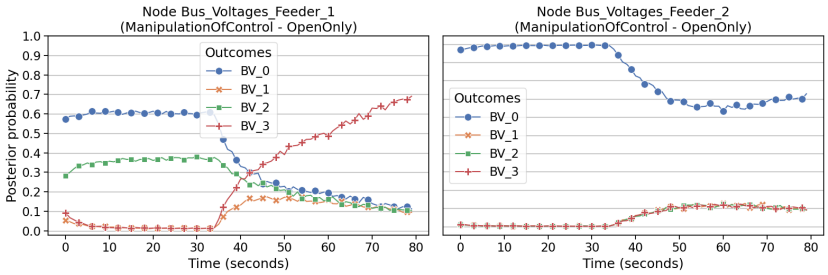
Figure 9: Results for the "No evidence" scenario.



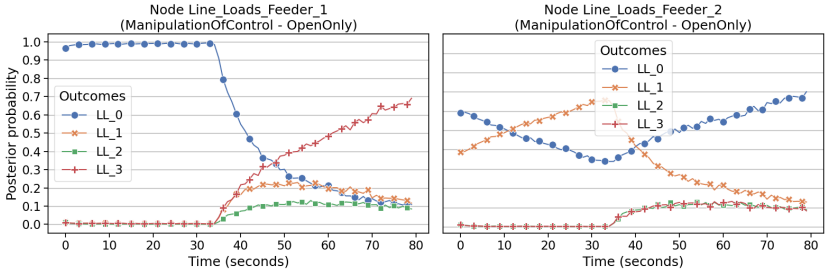
(a) Probability over time of outcome `C` for multiple attack step-related nodes.



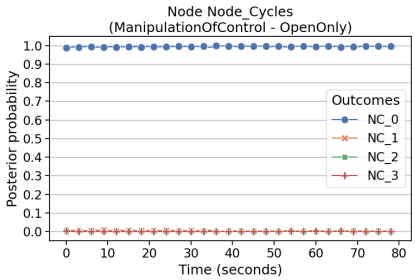
(b) Probability over time of the outcomes for node *ManipulationOfControl*.



(c) Probability over time of the outcomes for nodes *Bus_Voltages_Feeder_1* and *Bus_Voltages_Feeder_2*.

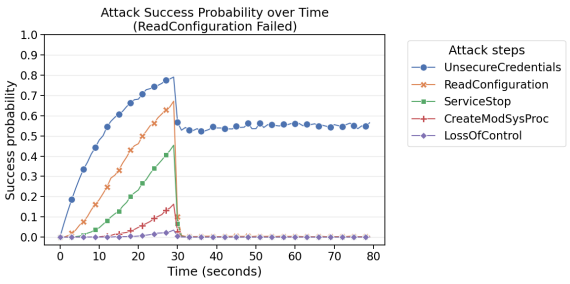


(d) Probability over time of the outcomes for nodes *Line_Loads_Feeder_1* and *Line_Loads_Feeder_2*.

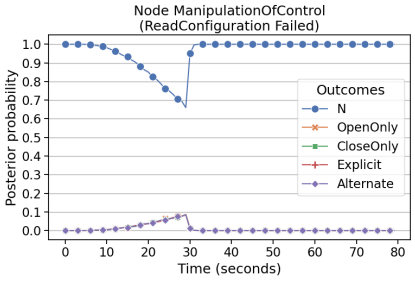


(e) Probability over time of the outcomes for node *Node_Cycles*.

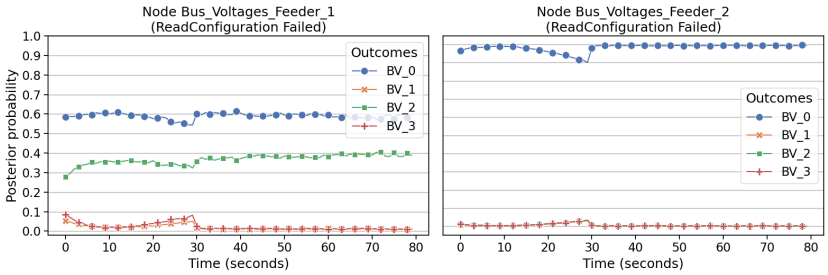
Figure 10: Results for the "ManipulationOfControl - OpenOnly" scenario.



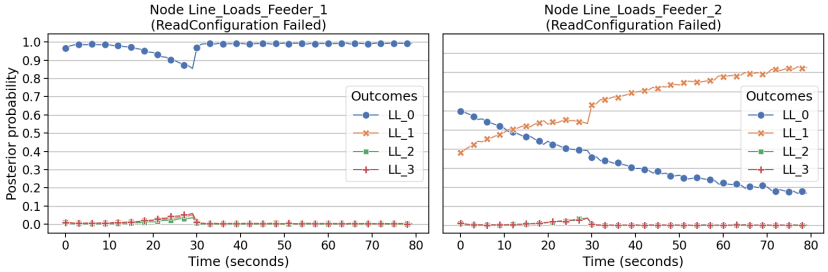
(a) Probability over time of outcome `C` for multiple attack step-related nodes.



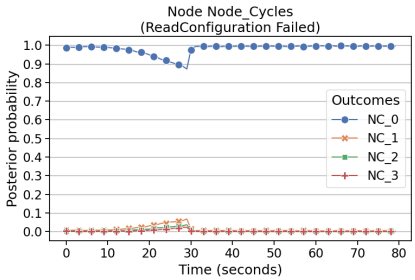
(b) Probability over time of the outcomes for node *ManipulationOfControl*.



(c) Probability over time of the outcomes for nodes *Bus_Voltages_Feeder_1* and *Bus_Voltages_Feeder_2*.

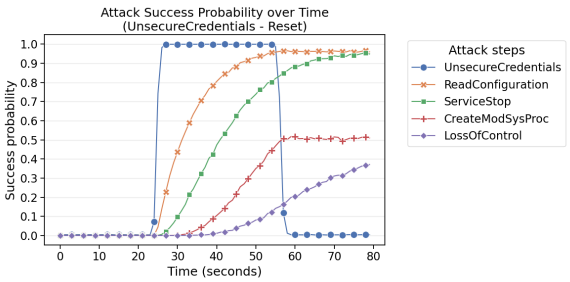


(d) Probability over time of the outcomes for nodes *Line_Loads_Feeder_1* and *Line_Loads_Feeder_2*.

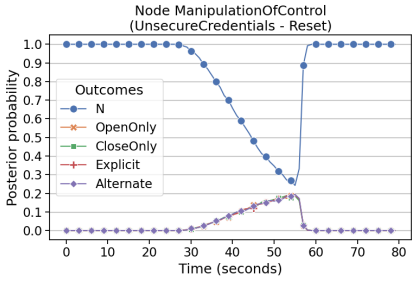


(e) Probability over time of the outcomes for node *Node_Cycles*.

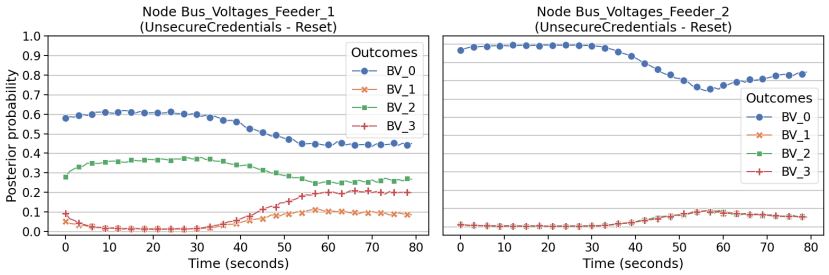
Figure 11: Results for the "ReadConfiguration - Failed" scenario.



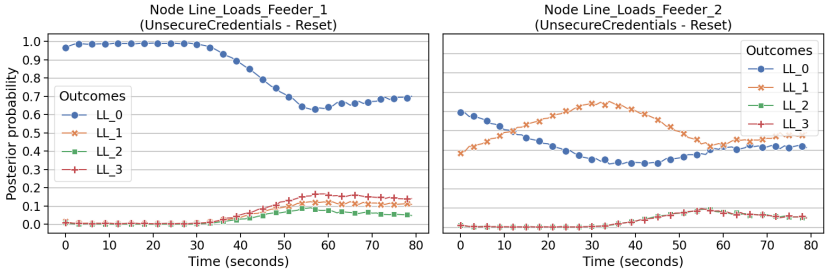
(a) Probability over time of outcome `C` for multiple attack step-related nodes.



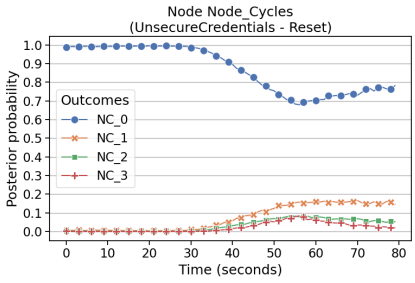
(b) Probability over time of the outcomes for node *ManipulationOfControl*.



(c) Probability over time of the outcomes for nodes *Bus_Voltages_Feeder_1* and *Bus_Voltages_Feeder_2*.



(d) Probability over time of the outcomes for nodes *Line_Loads_Feeder_1* and *Line_Loads_Feeder_2*.



(e) Probability over time of the outcomes for node *Node_Cycles*.

Figure 12: Results for the "UnsecureCredentials - Reset" scenario.

6.2 GSPN Inference Results

All the results presented in the following section were obtained by assuming the following to parameterize the GSPN as described in Section 5.2:

- All the evidences can be turned on or off in one unit of time (the rate associated to the relevant transitions would be then equal to one).
- All the timed transitions which controls the state changes within the power grid GSPN portion are weighted with a very low rate equals to 0.00000001. This assumption was made to propose an example of analysis focused only on the attack impact.
- The accuracy of all evidences is set to 0.1, except for *P_M1Det*, *P_M2Det*, *P_M3Det* and *P_cntLossDet1* (Figure 6 and 7) which are equals to 0.4.
- The timed transition associated to the steps completion are valued as in Table 9, while the timed transition associated to countermeasures are valued as in Table 10.

Unless otherwise specified, all the evidences are on[6] and the initial marking is:

```
Start=1, CounterNetAttack=2, NetWorking=1
```

Between the two alternatives presented in Section 3.4.1, unless otherwise is specified we used the model which consume tokens from evidence's on/off switches structure to improve the model checking performance. All the results were computed with GreatSPN [12] CSL^{TA} built-in solver (*MC4CSLTA*) configured with the following parameters value:

1. **Epsilon** equal to $1.0e-7$.
2. **Linear Solution** chosen is *GMRES*.
3. **MRP Method** *IMPLICIT*.
4. **Solution Method** as *Forward*.
5. **On the fly solution** checked.

6.2.1 Building DTAs to compute path properties

Unlike the DBN model, the expressiveness of GSPNs lies in the possibility of calculating the probability of all possible paths that satisfy certain properties. To do so, a query should refer to a sequence of different markings in GSPN model, whose evolution is timed and it is possible to express a precise temporal order. For example, the paths interested for a simple query such as "What's the probability the power grid crashed before Industroyer reaches the post-attack phase" could be seen in the GSPN as the ones where:

1. the GSPN is initialized as reported above, then
2. the *attackStarted* marking become equal to one, then

[6]This means to have a single token in each place of type "*EvidenceNameOn*" like *FileAccessUCOn* in figure 5. This can be done directly using a variable called *NEvidenceNameOn* associated to each evidence, such as *NFileAccessUCOn* in aforementioned figure.

Step in the AG	**Transitions in GSPN**	**Weight variable**	**Rate value**
UnsecureCredentials	$TUnsecureCred$	$UCRate$	0.5
ReadConfiguration	$TReadConfig$	$readRate$	1
ServiceStop	TServiceStop	$stopRate$	1
ManipulationOfControl	$TM_1 - TM_2 - TM_3$	$M_1Rate - M_2Rate - M_3Rate$	0.0167
CreateModSys	$TmodSys$	$modSysRate$	1
LossOfControl	$TLossCtrl$	$cntLossRate$	0.3
Time to reach *end*	$Tend$	$endRate$	0.1

Table 9: The table shows the rate value of all the timed transitions associated the attack steps.

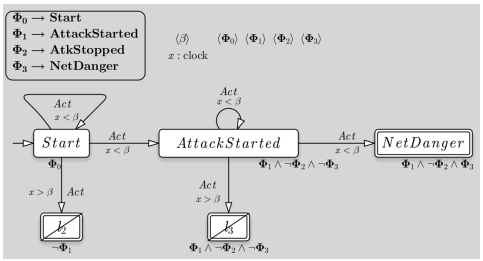
Evidence in the GSPN	**Countermeasure transition in GSPN**	**Rate value**
FileAccessUC	$TfileUCCNT$	0.5
FileAccess	$TfileCNT$	0.5
ProcessKill	$TStopCNT_1 - TStopCNT_2 - TStopCNT_3$	0.5
NetworkAnalysisAlarm	$TM_1CNT - TM_2CNT - TM_3CNT$	0.0167
WinRegKeyMod	$TmodSysCNT - TcntLossCNT_2$	0.5
FileIntegrity	$TcntLossCNT$	0.5

Table 10: The table shows the rate value of all the timed transitions associated the countermeasures which can stop the attack.

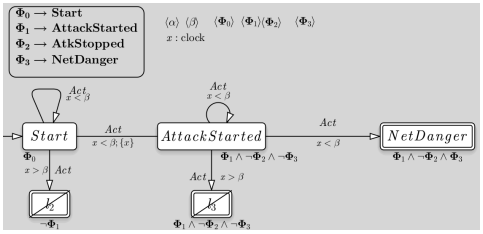
3. the *NetDanger* marking become equal to one, and finally
4. the *EnterPost* marking become equal to one.

It doesn't matter what happened between the steps above as long as each condition is met. The basic way to create a DTA (*cfr.* Section 2.2) to express similar query is to provide a different location for each change to the system status according to the query analyzed, as in the above enumeration. The transition from one location l_x to the next l_{x+1} is represented by a unidirectional edge that also specifies the time constraint within which the conditions of l_x changes to the conditions of l_{x+1}. Conditions can be expressed by a statement in the form of a logical formula containing parametrized logical variables. In GreatSPN, these variables must be associated to a place marking in the CSL^{TA} formula to apply the model checking. If the conditions of x_{x+1} are not satisfied within the specified temporal constraint associated to an edge, the DTA cannot accept the path. Figure 13 shows different DTAs created by following this idea. Queries expressed and results obtained with these DTAs will be discussed in the Section 6.2.2.

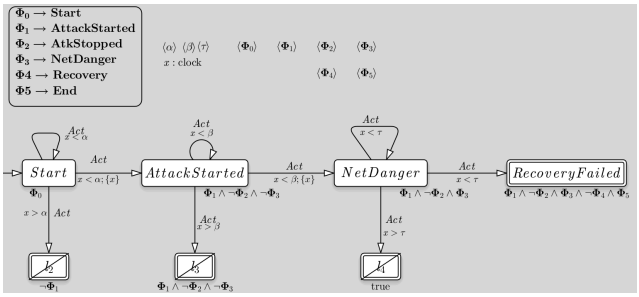
It is also interesting to note that a single automaton can be used to study many queries by only changing the logical variables association used in the state propositions on locations and the temporal variables used in the temporal constraint. For example, in the analysis proposed in Section 6.2.3 Φ_4 from the DTA in Figure 14a will be associated to the evidence $ProcessKill$, but it is sufficient to change this association in the CSL^{TA} formula to reuse the same DTA for the same query but on a different evidence.



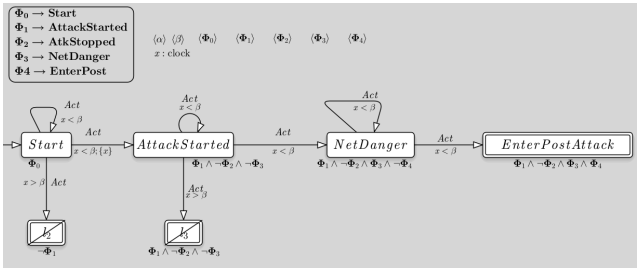
(a) DTA_1 is used for paths where the attack causes the power grid *danger* state without attack stopping in β units of time maximum.



(b) DTA_2 is similar to DTA_1, but the reset on the edge between *Start* and *AttackStarted* fixs the β limit only after the effective attack starting.



(c) DTA_3 specifies a complete path from the beginning to the end of GSPN, where accepted paths are the ones where both the Industroyer attack and the post-attack phases are successful without interruptions.



(d) DTA_4 which express paths were the power grid crashes before Industroyer enters in its post-attack phase. Such queries could be answered with DTA_1 and DTA_2 instead, becuase of the GSPN structure and the chosen parametrization the *NetDanger* location in DTA_4 is almost impossible to be reached.

Figure 13: Deterministic Timed Automata used in statistical model checking examples about attack effectiveness without any particular evidence trigger.

6.2.2 Computing the attack effectiveness without any specific evidence trigger

The first group of DTAs shown in Figure 13 was designed to analyze probability of all the different outcomes of the attack without any particular assumption on the evidence triggers. In these scenarios, the evidences can then be active or not, as they can be triggered or not.

The main objective is then to check the overall stability of the system if we suppose an Industroyer infection.

- DTA_1 (Figure 13a) and DTA_2 (Figure 13b) can be used to answer queries like: "what's the probability that the power grid crashes because of Industroyer in at least β units of time". To do so, the attack couldn't be stopped before it can influence the power grid, so Φ_2 must corresponds to the fact that *AtkStopped* in GSPN is never marked. Indeed, the final location accepts path where attack is started (GSPN place $AttackStarted == 1$ which is associated to Φ_1), attack is not stopped (GSPN place $AttackStopped! = 1$ which is associated to $\neg\Phi_2$) and the net is in *danger* (GSPN place $NetDanger == 1$ which is associated to Φ_3). The two DTAs can be distinguished by a clock reset on x labeling the edge between *Start* and *AttackStarted*. This reset leads to a slightly difference in the paths accepted: in the first DTA the time is considered from the starting marking of the model, while in the second the β units of time are counted from the effective attack starting. Before starting the model checking, it is interesting to note that one can answer the same queries with all, partially or fully without the monitoring system modeled in the GSPN without changing DTA_1 and DTA_2. Indeed, it is sufficient to change the initial marking of the GSPN removing all the tokens from the interested evidence on/off switch structure. To do so, it is important to use the GSPN model which not consume token as explained in Section 3.4.1; the more are the evidences excluded from the queries, the better is the model checking performance. Here's the CSL^{TA} formula used to compute the following results:

  ```
  PROB_TA > 0 DTA_1 (beta=50| |Phi0 = #Start==1, Phi1 = #AttackStarted
      ==1, Phi2 = #AtkStopped==1, Phi3 = #NetDanger==1)
  ```

  Results for different β value obtained using GreatSPN are shown in Table 11. The first two columns shows the case where the monitoring system is completely off: the probability increases as the time available for the attack increases, stabilizing after 400 time units (where a probability of 47% is reached) even with higher values of β. As expected, since there isn't a significant pre-attack phase in our GSPN model, there are few differences between the two DTAs results. Restoring the inital marking proposed in the previous section, model checking can be then done on a system with the monitoring system fully on, but without any particular evidences trigger fixed in the paths accepted. These results computed only with DTA_1 (because using DTA_2 results are almost the same) has been included in third column of Table 11.

  Finally, the two DTAs can be also used to investigate an opposite class of queries: "what's the probability Industroyer will be stopped without a power grid crash in at least in β units of time". To do so, it is sufficient to reverse the previous association of Φ_2 and Φ_3 in the CSL^{TA} formula shown above. Then, Φ_0 and Φ_1 remain the same while GSPN place $NetDanger == 1$ will be associated to Φ_2 and $AttackStopped == 1$ will be associated to Φ_3. The last column of Table 11 shows the results obtained for that query (with all the evidences on, since in the opposite case the probability will be zero as there is no possibility to stop the attack if the monitoring system is completely off).

- DTA_3 (Figure 13c) describes paths which represent the complete chain of Industroyer events, where the goals of both the attack and the post-attack phase must be reached without being stopped. In order to have an accepted path, after the attack initiation the path must stay in the location *AttackStarted* which represents the attack evolution

until the power grid crash (as for previous example, in the main tested scenario $\neg\Phi_2$ is basically $AtkStopped = 0$ and Φ_3 is $NetDanger = 1$). Once the power grid has crashed, its status must not change until the end of the simulation for the path to be accepted. The final location of acceptance is then reached if the *End* GSPN place is marked, which exactly corresponds to a successful end of the Industroyer post-attack phase (which can be also be underlined by the constraint on Φ_5 associated to the *End* place and Φ_4 which can be associated to *Recovery* place (as explained in Section 3.4.3 this place is the equivalent of *AttackStopped* in the attack phase modeled by the GSPN). Besides, this association on Φ_5 and Φ_4 allows us to study two different queries using the same DTA: the first case corresponds to reaching the end of the simulation without the recovery of the control room ($Phi5 = End == 1$ and $Phi4 = Recovery == 1$, the negation on Φ_4 in DTA implies that recovery failed); the second scenario is the opposite where the recovery ($\Phi_5 = End == 0$ and $\Phi_4 = Recovery == 0$, so the negation on Φ_4 in DTA implies that recovery was successful). There are three time variables which control all the DTA's constraints. α is the maximum amount of time allowed to complete the preliminary set up (such as turning on/off some evidences) or a pre-attack phase which is not modeled in the GSPN used. Because of that, the variation in α has very little effect on the results. It was therefore set to 10 just to keep a minimal impact in the performance of model checking. Similarly β and τ are the time allowed to reach the goals of Industroyer attack and post-attack phase. Each edge which represent accepted paths is labeled with a clock reset, so we can assume a precise maximum duration of each macro step. To test this DTA, we started focusing on different values of τ, since we have already explored probability of the attack phase completion using DTA_1: β will then be assumed equal to 300 units of time, as we derived before that with the monitoring system on the probability of a power grid crash is nearly 38% and stable for β values greater than 300. Results obtained for different value of τ and for both the scenario obtained by inverting the marking value of Φ_4 and Φ_5 are reported in Table 12. What we can find from model checking on this scenario is that for both scenario a τ value greater than 30 is sufficient to reach a point where the probability of path acceptance increases very slowly and tends to a fixed value, which is approximately 28% for scenario where there is no recovery of the control room and 9% in the opposite case. The rapid stabilization of the values was not surprising since the post-attack phase is composed of only two steps and rates parameterized are very low: $modSysRate = 1$, $cntLossRate = 0.2$, $endRate = 0.1$ and all the rates to reach the *Recovery* state because of a countermeasure are 0.5. It is interesting that for both DTA_1 and DTA_3, the chosen GSPN parametrization promotes successful attacks rather than interruptions and recovery. An example of a study on the effectiveness of an evidence with different GSPN weights will be provided in the Section 6.2.3.

- DTA_4 (Figure 13d) is a little extension of DTA_1 which formulates paths where the power grid crashes before Industroyer entering its post-attack phase. The associated class of query (such as "what's the probability that the power grid crashes before Industroyer is reaching the post-attack phase in at least β units of time") is particularly interesting because it cannot be studied with the DBN model. Even if queries and the DTA structure seem very similar than the previous point, model checking returns a probability extremely little (such as 1.353948E-6 for $\beta = 500$): this is because of the structure and the parameterization of GSPN. Indeed, in the net evolution the *enter-*

Post place is reached immediately after the end of the (looped) *ManipulationOfControl* by a series of immediate transition. In other words, in GSPN no time is passing between the end of *manipulationOfControl* and the state which tracks the post-attack initiation. Besides, because of the value established for the marking of *CNA* (which is two token) and the possible influence on the power grid (Section 3.4.1), the *danger* state can be reached only by the last *manipulationOfControl* cycle (as after two cycles *CNA* will be empty and so *enterPost* will be immediately initialized after the *manipulationOfControl* step completion). It is then virtually impossible to have $NetDanger$ marked and $EnterPost$ not marked in the same location, at least with the parameters chosen for this example, where the rates for autonomous changes in the power grid status are extremely low. To prove empirically that a device like DTA_4 cannot provide more information than DTA_1 and DTA_2 with this parametrization it is possible to remove $\neg\Phi_4$ from *NetDanger* location: for the same β value, results obtained with DTA_4 differ very slightly from those obtained with DTA_1 and DTA_2, at the level of sixth decimal place.

β	DTA_1 **(MonitoringOFF && !AttackStopped && NetDanger)**	DTA_2 **(MonitoringOFF && !AttackStopped && NetDanger)**	DTA_1 **(MonitoringON && !AttackStopped && NetDanger)**	DTA_1 **(MonitoringON && AttackStopped && !NetDanger)**
10	0.001515901629	0.002026469645	0.001394267708	0.077531519716
50	0.079412240671	0.082184683175	0.069458866296	0.115749998632
60	0.10777875769	0.110660400349	0.093607040065	0.123204824485
70	0.136715717982	0.139607490534	0.117992887967	0.130289892211
80	0.165377584319	0.168208292087	0.141925115466	0.136841728723
90	0.193165263409	0.195885358478	0.164932868852	0.142784523857
100	0.219670339773	0.222246948663	0.186710460627	0.148098908375
200	0.39397070295	0.394973803531	0.326123302095	0.175036480036
300	0.451984914516	0.452271489363	0.370959489223	0.180696806926
400	0.467661163648	0.467733555063	0.382865578839	0.181784243288
500	0.471509675505	0.471526786337	0.385759536765	0.18198991309
1000	0.472661128699	0.472661142159	0.386619105789	0.182037694012
2000	0.472666920082	0.472666925359	0.386623799877	0.182037704205

Table 11: Results obtained using DTA_1 and DTA_2, which model paths to answer the following: "what's the probability that the power grid crashes because of Industroyer at least in β units of time".

6.2.3 Computing the attack effectiveness with one evidence trigger

So far, scenarios have been proposed in which the attack can be stopped, but without imposing any evidence triggers along the path. Obviously, the inclusion of evidence in the analysis has several uses. For example, regardless of the outcome of the attack, the trigger of an evidence can represent a proof that the attack took place. Providing a probabilistic answer to this type of question can be done with the DBN model proposed above. In GSPNs, the model follows the steps of the attack exactly, so evidence can only be triggered by the attack. In other words, "false positives" have not been modeled. For this reason, when

τ	DTA_3 **NetDanger && !Recovery**	DTA_3 **NetDanger && Recovery**
15	0.184927650261	0.085991978222
20	0.222825702534	0.087698088649
25	0.246317078652	0.088081267646
30	0.26067773735	0.088166901374
35	0.269413000985	0.08818602584
40	0.274716812168	0.088190302884
50	0.279887224111	0.088191491039
75	0.282649843305	0.08819160218
100	0.282876737166	0.088191640269

Table 12: Results obtained using DTA_3 which has been used to model paths to answer the following: "what's the probability to reach the end of the attack with the power grid crashed because of Industroyer in at least β units of time and with/without recovery of the control room in at least τ units of time".

using the proposed GSPN, it makes no sense to ask "what is the probability of being under attack if evidence x has been triggered", because the probability will certainly be equal to one. Rather, the evidence trigger enriches the paths modeled by the DTAs, allowing the analyst to ask questions about the effectiveness of detection and defense of a certain evidence, comparing, for example, how the probability of attack success varies when the evidence is triggered. All previously submitted DTAs can be extended structurally or via state proposition to include the trigger of one or more evidences (as well as declaring the failure to trigger).

An example of basic analysis focused on a specific evidence can be done with $DTAev1$ (Figure 14a) which is structurally identical to DTA1 but has some variations in some edge clock guards and state propositions. With the Φ_x associations shown in the figure, the automaton could be used to answer the following question: "given that in at least α units of time the evidence *processKill* is triggered, what's the probability to stop the attack without a power grid crash in at least β units of time from the evidence trigger". we already know from the results obtained with DTA_1 in Section 6.2.2 that the probability to stop the attack (with the current parametrization introduced in Section 6.2) is around 18% (for an attack duration almost higher than 300 units of time). Now, $DTAev1$ can be used to study the effects of one evidence, assumed to be *ProcessKill*, on the same scenario. An example of CSL^{TA} formula in GreatSPN:

```
PROB_TA > 0 DTA_ev1 (alpha=10, beta=50| |Phi0 = #Start==1, Phi1 = #
    AttackStarted==1, Phi2 = #AtkStopped==1, Phi3 = #NetDanger==1, Phi4 = #
    ProcessKill==1)
```

ProcessKill was chosen because, according to the GSPN structure, it is triggered when the legit TCP service is stopped and can avoid the *manipulationOfControl* step, which is the only one that can be repeated more than once in the GSPN. To be aligned with the infor-

mation computed with DTA_1, the next analysis will be made with $\alpha = \beta$ and by removing the clock on the edge reset between l_0 and l_1. We'll then assume $\beta = 300$, and by computing the results we'll see that the probability is just under 5% (precisely 0.048012632027). This result may be counterintuitive, but model checking returns a probability calculated on all possible paths, accepting only those in which *ProcessKill* is activated and the attack is stopped. In other words, paths in which the attack is stopped before the *ProcessKill* trigger are not accepted[7], while paths where the attack is stopped thanks to the subsequent evidence, *networkMonitoring*, are accepted. Overall, the more constrained the path, the lower the probability compared to a path with the same outcome but without constraints. For this reason, the result obtained should rather be compared with paths that are equally constrained but with the opposite outcome, i.e., those paths in which, despite the *ProcessKill* trigger, the attack is not stopped and the power grid crashes. To do so, it is sufficient to revert Φ_2 and Φ_3 associations, as it was done in the previous section. By doing this, the result obtained is definitely lower than the opposite case: 0.000023552595, so the evidence role is not completely null, as the previous result would apparently suggest.

To study *ProcessKill* contribute in the GSPN execution, we then considered these two scenarios, both modeled with $DTAev1$, in order to explore how the probability changes according to the parameterization. The results are summarized in Table 13. An interesting thing is that variations on $P_{stopDet}$ seem to be much more effective than ones on $stopCMrate$, with a probability roughly ranging between 4% and 48% compared to a fixed value around 4.8% even for much higher $stopCMrate$ values than five. Of course, this last result it's tightly related to the rate associated to *ManipluationOfControl* step, which was set to 0.0167, but the impact of $P_{stopDet}$ value on results is still significant: for example, trying $P_{stopDet} = 1$ and $stopCMrate = 0.0001$ (which means that the time required to stop the attack is greater than the *ManipulationOfControl* step by a factor of 1000) the result is 0.052267101813 which is still higher than all results obtained with higher rate but a value of $P_{stopDet}$ equal to 0.1. One could argue that even if the attacks wouldn't be stopped by *ProcessKill* the probability of a power grid crash is still very low, however, if the attack is not stopped properly then it will continue with the post-attack phase which could end with a total control room disruption: definitely not a desirable state even if the grid is still functioning (or is in *alert* state). For *ProcessKill*, these proofs should be enough to convince IT department to invest in the accuracy of the evidence for this particular attack, rather than further increasing its countermeasures response time.

From the analysis, it would appear that, at least for *processKill*, response time is less important than the precision that regulates triggering. This observation makes perfect sense in the proposed GSPN model, given that each piece of evidence must first be triggered before it can trigger the countermeasure, which transition must then "compete" with the one that allows the attack to advance. Regardless of its speed, without a trigger of the associated evidence, there is no way for the countermeasure to stop the attack. The real impact of the speed of application of the countermeasure to stop the attack therefore risks being overshadowed by the probability of evidence triggering. It is possible to avoid this influence assuming that the evidence is already triggered and the attack is already at the relevant steps when we start the analysis. This can be easily done by changing the initial marking and adopting a reduced version of the DTA used until now: such as $DTAev1\_part$ (Figure 14b). In this analysis, associations for all Φ_x in $DTAev1\_part$ should remain the same, while the initial marking of GSPN must be changed by removing a token from *Start*

[7]To include these paths, it would be necessary to extend $DTAev1$ by adding a new final location with state proposition Φ_2 and reachable from l_0 if $\alpha < 0$

and *CounterNetAttacck* places, then setting $AttackStarted = 1$,$ProcessKill = 1$ and one of the $\_Ready = 1$ place of the GSPN portion which models the *ManipulationOfControl* step (*cfr.* Section 3.4.1), such as $OnlyOnReady$, $OnlyOffReady$ and $AlternateReady$ in Figure 6[8]. Table 14 provides all the results obtained for different $stopCMrate$ values and starting from $OnlyOnReady$ (due to the parameterization chosen there is no difference than starting from another $\_Ready$ place). Results show that a rate value equal to the attack step (which was set to 0.0167) is sufficient to reach a disappointing probability of 8% of stopping the attack: it's needed to speed the countermeasure approximately by a factor of twelve compared the actual *ManipulationOfControl* rate value to reach a probability greater than 60%. For this reason, the value equal to 0.5 chosen at the beginning of the section for these experiments is certainly excellent, given that it implies a probability of over 90% of stopping the attack.

6.2.4 Comparing all the evidence effectiveness

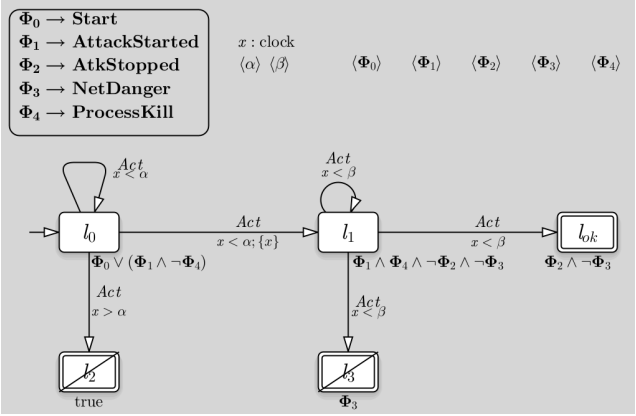
As the previous section discussed, the analysis focused on the evidence *ProcessKill* suggests that investing in the accuracy of this evidence could be useful, since as reported in table13 probability of stopping the attack ranges between a minimum of 0.4 in the chosen configuration to a maximum of approximately 0.48. However, the positive result couldn't be enough without comparing how other evidence protect the system against the attack. A simple analysis can be performed by using $DTAev1$ with the same configuration and logic described in section 6.2.3 to study *ProcessKill*. The chart in figure 15b shows the trend of the probability of stopping the attack for each evidence involved in the Industroyer attack phase.[9] Each value was computed exactly with the initial configuration of the GSPN (except for the specific probability associated to the studied evidence) and $DTAev1$'s temporal variables fixed as previous section: $\alpha = \beta = 300$. By comparing all the evidence in this way, we can finally conclude that investing in the accuracy of *ProcessKill* should be the best choice, as the chances of stopping the attack increases more than any other evidence accuracy variation. Of course, using $DTAev1$ to analyze the system is a bit unfair, especially for the latest evidence in the model, because the path described by the DTA includes a mandatory evidence trigger before a final state acceptation and, at the same time, DTA doesn't accept any paths where the attack is stopped before this evidence trigger. In this way, given a single evidence studied, the probability of stopping the attack after this evidence trigger also benefits from any other following evidence that can stop the attack even after the one studied.

[8] A good alternative could be to set a token in $EndS_3$ instead of $\_Ready$. This decision has less changes than the initial marking provided in Section 6.2, but to perform CSL^{TA} model checking in GreatSPN it is necessary to have a single initial state (which isn't true starting from End_3 since is connected to immediate transitions).

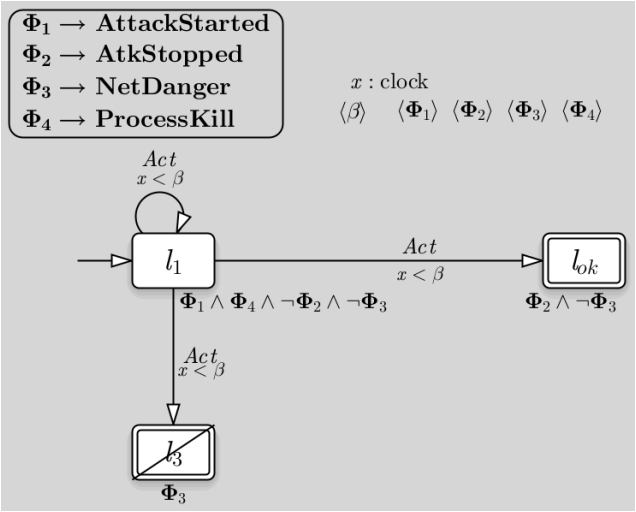
[9] Given that the trend of the three*ManipluationOfControls* evidence is practically identical (differences are visible only after the fifth decimal place) figure 15b displays only a single line for them.

		$DTAev1$ **NetDanger && !AttackStopped**	$DTAev1$ **!NetDanger && AttackStopped**
$stopCMrate$	0.0001	0.019406444173	0.005226710288
	0.001	0.017677515766	0.009297588298
	0.0167	0.036847996000	0.036847996
	0.1	0.000453042000	0.047103967774
	0.5	**0.000023552595**	**0.048012632027**
	1	0.000006101842	0.048049551985
	2	0.000001553414	0.04805917485
	5	0.000000251341	0.048061929600
$P\_stopDet$	**0.1**	**0.000023552595**	**0.048012632027**
	0.2	0.000047105190	0.096025263625
	0.3	0.000070657784	0.144037894794
	0.4	0.000094210378	0.192050525532
	0.5	0.000117762972	0.240063155842
	0.6	0.000141315566	0.288075785721
	0.7	0.000164868160	0.336088415171
	0.8	0.000188420753	0.384101044191
	0.9	0.000211973346	0.432113672782
	1	0.000235525939	0.480126300942

Table 13: Results obtained using $DTAev1$ starting from the parametrization described in section 6.2. First seven rows shows the results by keeping $P_{stopDet} = 0.1$ and changing the rate associated to the transitions that leads to *AttackStopped*. Next rows, $P_{stopDet}$ (the probability that the evidence *ProcessKill* is triggered) is varied, while $stopCMrate$ is fixed to 0.5.



(a) $DTAev1$ express paths where an evidence (for example *ProcessKill*) is triggered during the attack which is then stopped.

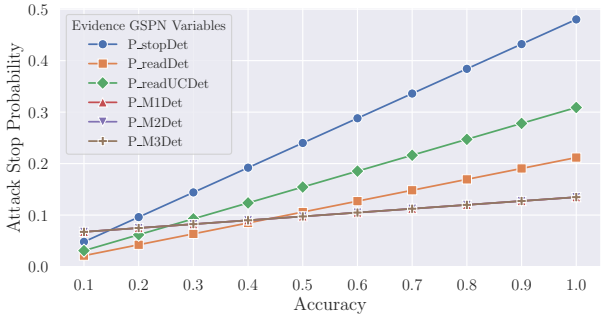


(b) $DTAev1\_part$ is a reduced version of $DTAev1$ used to queries the model with a different marking than the basic initial one declared in section 6.2.

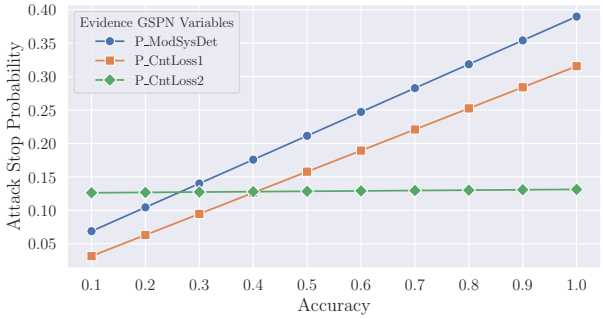
Figure 14: Deterministic Timed Automata used in statistical model checking examples about attack effectiveness assuming at least an evidence trigger.

$stopCMrate$	$DTAev1\_part$ **NetDanger && !AttackStopped**	$DTAev1\_part$ **!NetDanger && AttackStopped**
0.0001	0.001486006854	0.001272442287
0.001	0.001481651253	0.005754055786
0.0167	0.001407949400	0.080775384100
0.03	0.001348757622	0.139880961434
0.05	0.001264999976	0.221682970647
0.1	0.001080396048	0.393741565343
0.2	0.000796916310	0.632131419995
0.3	0.000596900928	0.776753937653
0.4	0.000454158303	0.864495909187
0.5	**0.000351056943**	**0.917732408976**
1	0.000121294120	0.993151003247
2	0.000032278819	0.999921935337

Table 14: Results obtained using $DTAev1part$. The automaton is meant to study more precisely the effectiveness of the countermeasures associated to the *ProcessKill* evidence, assumed as already triggered, starting the analysis in the middle of the attack (before the *ManipulationOfControl* step).



(a) The graph shows the evolution of the probability of stopping the attack by increasing the accuracy of each evidence involved in the attack phase.



(b) The graph shows the evolution of the probability of recovering the system by increasing the accuracy of each evidence involved in the post attack phase.

Figure 15: Results gathered by using $DTAev1$ as in the section 6.2.3 on all the evidence of the GSPN model.

7 Related work

The digitization process of energy systems we witnessed during the last decade, led to the development of more resilient and efficient generation, transmission and distribution infrastructures, accommodating the need of flexibly managing facilities of Renewable Energy Resources (RERs). This abrupt development poses new challenges for securing such decentralized monitoring and control infrastructure where the legacy Operational Technology (OT) devices integrate with more modern Information Technology (IT) derived components. With these premises, we compare our security assessment methodology to alternative solutions employed in already existing frameworks.

One of the most relevant security assessment frameworks is ADVISE (ADversary VIew Security Evaluation) [22, 35]. It is based on designing an Attack Execution Graph (AEG), which takes 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. Considering also the characteristics of specific adversaries, including their cost-aversion, risk-aversion and available time to protract the attack, a State LookAhead Tree (SLAT) is generated. By traversing this data structure, the security analyst could gather deeper insights on the paths that the attacker may pursue or to obtain measures that provide a quantitative assessment of possible consequences of the adversarial activity. While ADVISE offers a more extensive attacker modeling solution with the possibility of modifying his behavior based on some predefined parameters, the parameterization process still remains a major burden for the final user and it could become a challenging task when technical domain knowledge is necessary. Additionally the SCADA network case study presented in the original work do not attempt to model any power-related phenomena, but mainly focuses on analyzing the vulnerabilities from a high level architectural perspective. Our inference models are instead parameterized exploiting realistic attack information to learn the impact of specific attack strategies on the power grid model. Moreover, while ADVISE strictly relies on start-to-goal offline attack analysis, providing metrics regarding the expected time for the attacker to reach his desired goal, our methodology can integrate real time evidence of the attack steps performed and output probabilistic information on the state of each attack step or on the state of the power grid over time, using the DBN model. Additionally the GSPN model, using its statistical model checking capabilities, can provide probabilistic insights on the satisfiability of more complex temporal properties declared on events paths which include both the attack steps and the evidences influence on the attack success. An interesting extension of ADVISE is the Advise Meta Modeling Framework [20, 19], which creates a higher level abstraction 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). Apart from possibly facilitating the parametrization and extension process, as it was attempted in [24], this formalism does not solve the other shortcomings of the original ADVISE framework we already mentioned.

Another noteworthy framework is the Meta Attack Language (MAL) [46], which provides the tools to define a Domain Specific Language (DSL) to model the possible attack steps, the possible defenses and the means by which 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. While ours is still a MAL-derived AG, we integrated real world analytics to analyze how the attack steps evolve over time.

Up until this point we have been focusing on more general purpose security assessment frameworks that did not attempt to model the relation between cyberattacks and power flow stability, but we will now explore two examples of such tools.

In [17] a risk assessment framework for CPPSs in distribution grids was presented, modeling Distributed Energy Resources (DERs) through three layers: control, physical system, and communication. The control layer applies algorithms such as Optimal Power Flow (OPF) and load-sharing, while potential vulnerabilities are identified through the National Vulnerability Database (NVD), and the most common ones are modeled using analytical methods. Impact quantification combines the likelihood of an attack with its potential impact expressed in terms of financial losses and system stability degradation. Validation was performed through simulations on modified IEEE 13-node and 123-node test feeders, demonstrating its capability to identify vulnerabilities and assess risks in DER-integrated distribution grids. While this framework analyzes system responses to attack scenarios individually, our framework is designed to support multi-stage cyberattack security assessment, enabling the modeling of more complex interdependencies within the cyber kill chain. Moreover, although [17] approach employs BNs to estimate the probability of attack success for each vulnerable node, it does not model the temporal evolution such our DBN does, nor it can answer path-based temporal properties like our GSPN model does through its statistical model checker.

In [47] a dynamic risk assessment model for CPPSs is proposed, focusing the network security vulnerabilities of substation monitoring and control infrastructure, as well as physical consequences that the malicious exploitation of such vulnerabilities could have. In this framework the probability of successful exploitation is computed starting from the features extracted from the Common Vulnerability Scoring System (CVSS) and its is integrated with additional information related to disclosure time and attacker's characterization. Combining this information a Probabilistic Attribute attacking-path Graph is defined and the maximum likelihood path from the initially exploited vulnerability to the target goal is computed. This model also estimates physical consequences of cyberattacks by simulating the activity of a load shedding mechanism in the power system, leading to tripping transmission lines and generators and possibly resulting in cascading failures of the entire network. While the protection system is not modeled in our emulation framework, [47] only considers the best path leading the attacker to his predefined goal based on the precomputed attack step probabilities. Our framework instead allows to evaluate multi-path attack strategies, computing attack success probabilities (based on mean TTC/I information), likelihood of specific power grid states over time and analyzing temporal probabilistic properties.

In general both [17, 47] put less effort in modeling attack interdependencies and their probabilities are less detailed and articulated compared to our approach, making our studies somewhat complementary in this sense.

As explored in [18, 10], the demand for dynamic risk assessment tools has motivated the adoption of BNs, whose capability of adjusting prior failure probabilities and immediately update the results makes them a suitable solution to be deployed in heterogeneous environments. However, all the mentioned works do not derive the BN starting from a higher-level model which could be more familiar for security analysts, as we detailed in section 3.3. Furthermore, none of these studies investigates the additional functionalities introduced by DBNs, which by incorporating temporal dependencies, support the early detection and the fast counteraction to attacks in real-time analysis, as we explored in section 6.1.

Finally, the approach presented in [31], referred to as DETECT, implements an Intrusion Detection System (IDS) that is relevant to our work as it employs BNs derived from Attack

Trees to identify evolving attacks. Within a predetermined time window, events are collected and monitored, and the resulting observations are used to update the BN parameters. The model is resolved after each event-driven update, potentially issuing an attack warning or alarm. Similarly, in our work DBNs and GSPNs are partially derived from AGs. A key difference is that that these models inherently incorporate a temporal dimension, enabling the analysis of attack step evolution over time through the use of temporal evidence and deterministic markings, respectively.

8 Conclusions

This work presented a complete workflow to assess cybersecurity risks in cyber-physical power distribution networks by converting an analyst-friendly AG representation to two complementary probabilistic formalisms: DBNs and GSPNs. The parameterization of this pipeline is based and tested on a realistic co-simulation of the Industroyer 1 cyber kill chain on the Wattson framework running the "CIGRE MV all DER" scenario and it uses the MITRE ATT&CK ontology to classify and structure attack steps and analytics. This allowed us to capture both the ICT-related attack fingerprints and their effects on the power grid model, maintaining a safe and reproducible experimental setup. On the DBN we mapped techniques and analytics into temporal and non temporal nodes and introduced additional logical nodes to adequately model the mix of persistent and non persistent attack steps. Additionally we modeled a power grid related section of the DBN and learned its CPTs from emulation traces via the EM algorithm. We also address the discrete time parameterization problem deriving success probabilities for each attack step starting from their mean TTC/I through the uniformization technique. This preserves the specified TTC/I means while enabling filtering analysis with streams of evidences. The result is a model that can compute posterior probabilities for both cyberattack steps and power grid metrics. Then, for the GSPN we chose to follow an approach that focused more on the attack and its steps, rather than on a detailed and expressive description of the power grid. Therefore, this model is also derived from the attack graph, but it describes better the ordered sequence of steps of the entire Industroyer attack, separating the temporal aspect from the probabilistic one to include in the analysis the idea that the trigger of an evidence can activate a countermeasure that stops the attack. In this way, in addition to allowing the study of temporal and probabilistic relationships between modeled events, the GSPN adds the possibility of studying separately the impact of the accuracy of each evidence and the conflict between the continuation of the attack and its mitigation, which depends entirely on the time required to complete the actions modeled. Our DBN scenarios illustrate how the parameterization process and the model structure drives the posterior probabilities when no evidence is provided, when attack alarms are raised and when negative evidences are present. They also demonstrate how partial or complete attack persistence and multi-strategy attack behavior shape technique success likelihoods and observable grid impact. Instead, in our GSPN scenarios the focus is on event paths where Industroyer surely infected the SCADA control room and the main investigations were made on relationship between the success of the attack (or its mitigation) and the status of monitoring/defense system, composed of all the evidence included in the attack graph and their supposed countermeasures against the associated attack step. After discussing how probabilities vary when all evidences are either off or on, we decided to show how the model can be used to analyze the contribution of a chosen evidence in mitigating the attack, comparing the results obtained by varying the

probability of the evidence triggering (i.e., its accuracy in identifying the associated attack step) and the time needed to interrupt the attack at the identified step. Some limitations of our work regard the utilization of emulated traces to train our DBN and GSPN models as real world data on this kind of cyberattacks is not yet openly available. Additionally power protection schemes are abstracted as we mainly focused on isolating attack dynamics.

As a future work we plan to:

1. expand our attack topologies beyond Industroyer-like campaigns.
2. validate our models against real-world datasets when available.
3. introduce data protection modeling and analyze its impact on power grid measurements.
4. integrate our security assessment models with real-time evidence provided by AI-based detection models.

These steps aim at providing an explainable, and temporally aware assessment capabilities for power distribution networks under active cyber threats.

References

[1] Marco Ajmone Marsan, Gianni Conte, and Gianfranco Balbo. "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems". In: *ACM Trans. Comput. Syst.* 2.2 (May 1984), pp. 93–122. ISSN: 0734-2071. DOI: 10.1145/190.191. URL: https://doi.org/10.1145/190.191.

[2] Elvio Gilberto Amparore, Marco Beccuti, and Susanna Donatelli. "(Stochastic) Model Checking in GreatSPN". In: *Application and Theory of Petri Nets and Concurrency*. Ed. by Gianfranco Ciardo and Ekkart Kindler. Cham: Springer International Publishing, 2014, pp. 354–363. ISBN: 978-3-319-07734-5.

[3] Elvio Gilberto Amparore and Susanna Donatelli. "Model checking CSLTA with Deterministic and Stochastic Petri Nets". In: *2010 IEEE/IFIP International Conference on Dependable Systems & Networks (DSN)*. 2010, pp. 605–614. DOI: 10.1109/DSN.2010.5544425.

[4] Lennart Bader et al. "Comprehensively Analyzing the Impact of Cyberattacks on Power Grids". In: *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*. 2023, pp. 1065–1081. DOI: 10.1109/EuroSP57164.2023.00066.

[5] BayesFusion, LLC. *SMILE: Structural Modeling, Inference, and Learning Engine*. Version 2.2.4.R1 (built April-27-2024). Cited according to developer recommendation for academic use. 2025. URL: https://www.bayesfusion.com/smile/.

[6] Xavier Boyen and Daphne Koller. "Tractable inference for complex stochastic processes". In: *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 33–42. ISBN: 155860555X.

[7] D. Cerotti et al. *A modular infrastructure for the validation of cyberattack detection systems*. Tech. rep. TR-INF-2022-05-01-UNIPMN. Computer Science Institute, UPO, 2022. URL: https://www.di.unipmn.it/en/publications-en/technical-reports-en.html?pubid=567.

[8] Davide Cerotti et al. "A Modular Infrastructure for the Validation of Cyberattack Detection Systems". In: *Power Systems Cybersecurity: Methods, Concepts, and Best Practices*. Ed. by Hassan Haes Alhelou, Nikos Hatziargyriou, and Zhao Yang Dong. Cham: Springer International Publishing, 2023, pp. 311–336. ISBN: 978-3-031-20360-2. DOI: 10.1007/978-3-031-20360-2_13. URL: https://doi.org/10.1007/978-3-031-20360-2%5C_13.

[9] Davide Cerotti et al. "SecuriDN: A Modeling Tool Supporting the Early Detection of Cyberattacks to Smart Energy Systems". In: *Energies* 17.16 (2024). ISSN: 1996-1073. DOI: 10.3390/en17163882. URL: https://www.mdpi.com/1996-1073/17/16/3882.

[10] Pavlos Cheimonidis and Konstantinos Rantos. "Dynamic Risk Assessment in Cybersecurity: A Systematic Literature Review". In: *Future Internet* 15.10 (2023). ISSN: 1999-5903. DOI: 10.3390/fi15100324. URL: https://www.mdpi.com/1999-5903/15/10/324.

[11] Jian Cheng and Marek J. Druzdzel. "AIS-BN: an adaptive importance sampling algorithm for evidential reasoning in large Bayesian networks". In: *J. Artif. Int. Res.* 13.1 (Oct. 2000), pp. 155–188. ISSN: 1076-9757.

[12] G. Chiola et al. “GreatSPN 1.7: Graphical editor and analyzer for timed and stochastic Petri nets”. In: *Performance Evaluation* 24.1 (1995). Performance Modeling Tools, pp. 47–68. ISSN: 0166-5316. DOI: https://doi.org/10.1016/0166-5316(95)00008-L. URL: https://www.sciencedirect.com/science/article/pii/016653169500008L.

[13] Sabarathinam Chockalingam et al. “Bayesian Network Models in Cyber Security: A Systematic Review”. In: *Secure IT Systems*. Ed. by Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevičius. Cham: Springer International Publishing, 2017, pp. 105–122. ISBN: 978-3-319-70290-2.

[14] International Electrotechnical Commission. *Telecontrol equipment and systems - Part 5-104: Transmission protocols - Network access for IEC 60870-5-101 using standard transport profiles*. This is the second edition of the standard. It supersedes the first edition published in 2003. International Electrotechnical Commission, 2006.

[15] Susanna Donatelli, Serge Haddad, and Jeremy Sproston. “Model Checking Timed and Stochastic Properties with CSLTA”. In: *IEEE Transactions on Software Engineering* 35.2 (2009), pp. 224–240. DOI: 10.1109/TSE.2008.108.

[16] Arnaud Doucet et al. “Rao-blackwellised particle filtering for dynamic Bayesian networks”. In: *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*. UAI’00. Stanford, California: Morgan Kaufmann Publishers Inc., 2000, pp. 176–183. ISBN: 1558607099.

[17] Xue Gao, Mazhar Ali, and Wei Sun. “A Risk Assessment Framework for Cyber-Physical Security in Distribution Grids with Grid-Edge DERs”. In: *Energies* 17.7 (2024). ISSN: 1996-1073. DOI: 10.3390/en17071587. URL: https://www.mdpi.com/1996-1073/17/7/1587.

[18] Priscilla Grace George and V.R. Renjith. “Evolution of Safety and Security Risk Assessment methodologies towards the use of Bayesian Networks in Process Industries”. In: *Process Safety and Environmental Protection* 149 (2021), pp. 758–775. ISSN: 0957-5820. DOI: https://doi.org/10.1016/j.psep.2021.03.031. URL: https://www.sciencedirect.com/science/article/pii/S0957582021001452.

[19] Ken Keefe et al. “An Ontology Framework for Generating Discrete-Event Stochastic Models”. In: *Computer Performance Engineering*. Ed. by Rena Bakhshi et al. Cham: Springer International Publishing, 2018, pp. 173–189. ISBN: 978-3-030-02227-3.

[20] Ken Keefe et al. “Enterprise Security Metrics with the ADVISE Meta Model Formalism”. In: *Proceedings of SECURWARE 2015: The Ninth International Conference on Emerging Security Information, Systems and Technologies*. Ed. by Rainer Falk, Carla Merkle Westphall, and Hans-Joachim Hof. Best Paper Award, archived in ThinkMind Digital Library. Venice, Italy: IARIA / ThinkMind, 2015, pp. 65–66. ISBN: 978-1-61208-427-5. URL: https://www.perform.illinois.edu/Papers/USAN_papers/15FED02.pdf.

[21] Uffe Kjærulff. “dHugin: a computational system for dynamic time-sliced Bayesian networks”. In: *International Journal of Forecasting* 11.1 (1995). Probability Forecasting, pp. 89–111. ISSN: 0169-2070. DOI: https://doi.org/10.1016/0169-2070(94)02003-8. URL: https://www.sciencedirect.com/science/article/pii/0169207094020038.

[22] Elizabeth LeMay et al. “Model-based Security Metrics Using ADversary VIew Security Evaluation (ADVISE)”. In: *2011 Eighth International Conference on Quantitative Evaluation of SysTems*. 2011, pp. 191–200. DOI: 10.1109/QEST.2011.34.

[23] Chih-Yuan Lin et al. “RICSel21 Data Collection: Attacks in a Virtual Power Network”. In: *2021 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. 2021, pp. 201–206. DOI: 10.1109/SmartGridComm51999.2021.9632328.

[24] Francesco Mariotti. “Cybersecurity Assessment: Challenges and Advancements around the ADVISE Meta Framework”. Dottorato di Ricerca in Matematica, Informatica e Statistica, Curriculum in Informatica, Ciclo XXXVII. PhD thesis. University of Florence, 2025. URL: https://flore.unifi.it/bitstream/2158/1416813/1/Thesis%20Francesco%20Mariotti.pdf.

[25] M. Ajmone Marsan et al. “Modelling with Generalized Stochastic Petri Nets”. In: *SIGMETRICS Perform. Eval. Rev.* 26.2 (Aug. 1998), p. 2. ISSN: 0163-5999. DOI: 10.1145/288197.581193. URL: https://doi.org/10.1145/288197.581193.

[26] Petr Matoušek, Ondřej Ryšavý, and Peter Grofčík. *ICS Dataset for Smart Grid Anomaly Detection*. 2022. DOI: 10.21227/1trw-n685. URL: https://dx.doi.org/10.21227/1trw-n685.

[27] Alessio Misuri et al. “A Bayesian network methodology for optimal security management of critical infrastructures”. In: *Reliability Engineering & System Safety* 191 (2019), p. 106112. ISSN: 0951-8320. DOI: https://doi.org/10.1016/j.ress.2018.03.028. URL: https://www.sciencedirect.com/science/article/pii/S0951832017311535.

[28] Kevin Murphy. “Dynamic Bayesian Networks: Representation, Inference and Learning”. PhD thesis. The University of British Columbia, Jan. 2002.

[29] V.F. Nicola, P. Heidelberger, and P. Shahabuddin. “Uniformization and exponential transformation: Techniques for fast simulation of highly dependable non-Markovian systems”. In: *[1992] Digest of Papers. FTCS-22: The Twenty-Second International Symposium on Fault-Tolerant Computing*. 1992, pp. 130–139. DOI: 10.1109/FTCS.1992.243607.

[30] pandapower developers. *Medium Voltage Distribution Network with all DER — pandapower v2.0.0*. https://pandapower.readthedocs.io/en/v2.0.0/networks/cigre.html. Accessed: 2025-08-04. 2020.

[31] Mauro José Pappaterra and Francesco Flammini. “Bayesian Networks for Online Cybersecurity Threat Detection”. In: *Machine Intelligence and Big Data Analytics for Cybersecurity Applications*. Ed. by Yassine Maleh et al. Cham: Springer International Publishing, 2021, pp. 129–159. ISBN: 978-3-030-57024-8. DOI: 10.1007/978-3-030-57024-8_6. URL: https://doi.org/10.1007/978-3-030-57024-8_6.

[32] Simon Parsons. “Jensen Finn V., Nielsen Thomas D. 2007. Bayesian Networks and Decision Graphs,Second EditionSpringer Verlag, 447 pp. ISBN 0-387-68281-3”. In: *The Knowledge Engineering Review* 23.4 (2008), pp. 413–413. DOI: 10.1017/S0269888908000076.

[33] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988. ISBN: 1558604790.

[34] M. Peuster, H. Karl, and S. van Rossem. "MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments". In: *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. Nov. 2016, pp. 148–153. DOI: 10.1109/NFV-SDN.2016.7919490.

[35] Michael Rausch et al. "A Comparison of Different Intrusion Detection Approaches in an Advanced Metering Infrastructure Network Using ADVISE". In: *Proceedings of the 13th International Conference, QEST 2016*. Vol. 9826. Aug. 2016, pp. 279–294. ISBN: 978-3-319-43424-7. DOI: 10.1007/978-3-319-43425-4_19.

[36] Luis Salazar et al. "A Tale of Two Industroyers: It was the Season of Darkness". In: *2024 IEEE Symposium on Security and Privacy (SP)*. 2024, pp. 312–330. DOI: 10.1109/SP54263.2024.00162.

[37] Joe Slowik. "Anatomy of an Attack: Detecting and Defeating CRASHOVERRIDE". In: *Proceedings of Virus Bulletin Conference (VB2018)*. Available via Virus Bulletin website. Montreal, Quebec, Canada, Oct. 2018. URL: https://www.virusbulletin.com/virusbulletin/2019/03/vb2018-paper-anatomy-attack-detecting-and-defeating-crashoverride/.

[38] K. Strunz and Stefano Barsali. "CIGRE Task Force C6. 04.02: Developing Benchmark Models for Integrating Distributed Energy Resources". In: *Power Engineering Society General Meeting* (Jan. 2005).

[39] The MITRE Corporation. *2015 Ukraine Electric Power Attack*. https://attack.mitre.org/campaigns/C0028/. 2024.

[40] The MITRE Corporation. *2016 Ukraine Electric Power Attack*. https://attack.mitre.org/campaigns/C0025/. 2025.

[41] The MITRE Corporation. *Adversarial Tactics, Techniques and Common Knowledge (ATT&CK)*. https://attack.mitre.org/. 2015.

[42] The MITRE Corporation. *ATT&CK Data Sources*. https://attack.mitre.org/datasources/. 2015.

[43] The MITRE Corporation. *ATT&CK for Enterprise*. https://attack.mitre.org/matrices/enterprise/. 2015.

[44] The MITRE Corporation. *ATT&CK for Industrial Control Systems*. https://attack.mitre.org/matrices/ics/. 2020.

[45] Leon Thurner et al. "Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems". In: *IEEE Transactions on Power Systems* 33.6 (2018), pp. 6510–6521. DOI: 10.1109/TPWRS.2018.2829021.

[46] Wojciech Wideł et al. "The meta attack language - a formal description". In: *Computers & Security* 130 (2023), p. 103284. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2023.103284. URL: https://www.sciencedirect.com/science/article/pii/S0167404823001943.

[47] Kang Yan et al. "A Cyber-Physical Power System Risk Assessment Model Against Cyberattacks". In: *IEEE Systems Journal* 17.2 (2023), pp. 2018–2028. DOI: 10.1109/JSYST.2022.3215591.

[48] Changhe Yuan and Marek J. Druzdzel. “An importance sampling algorithm based on evidence pre-propagation”. In: *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*. UAI'03. Acapulco, Mexico: Morgan Kaufmann Publishers Inc., 2002, pp. 624–631. ISBN: 0127056645.