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

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 (davide.cerotti@uniupo.it, daniele.codetta_raiteri@uniupo.it, giovanna.dondossola@rse-web.it, lavinia.egidi@uniupo.it, giuliana.franceschinis@uniupo.it, luigi.portinale@uniupo.it, roberta.terruggia@rse-web.it)

TECHNICAL REPORT TR-INF-2022-05-01-UNIPMN
(May 2022)

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

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.

2018-02 *Power (set) Description Logic*, L. Giordano, A. Policriti, February 2018.

2018-01 *A Game-Theoretic Approach to Coalition Formation in Fog Provider Federations (Extended Version)*, C. Anglano, M. Canonico, P. Castagno, M. Guazzone, M. Sereno, February 2018.

2017-02 *Configuration and Use of Android Virtual Devices for the Forensic Analysis of Android Applications (see below for citation details)*, C. Anglano, M. Canonico, M. Guazzone, June 2017.

2017-01 *A dynamic simulation model for comparing kidney exchange policies*, M. Beccuti, G. Franceschinis, S. Villa, March 2017.

A modular infrastructure for the validation of cyberattack detection systems

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

[1] UdR UPO, Consorzio Nazionale Interuniversitario per le Telecomunicazioni, Italy
[2] DiSIT, Università del Piemonte Orientale (UPO), Alessandria, Italy
[3] Transmission and Distribution Technologies Department, Ricerca sul Sistema Energetico, Milano, Italy
May 2022

Abstract

We propose a framework for the evaluation of cyberattack detection systems in which methodological results can be tested in a realistic setup. We emulate a power control infrastructure, an attacker and a monitoring systems. In this controlled environment, through a modular approach, it is possible to evaluate a variety of detection models: we inject adversarial activity, collect logs from the systems, analyze such logs and produce evidences that are later processed by artificial intelligence models that can raise alerts, and give diagnostic or predictive information. The testbed allows us to effectively test the adequacy of the detection mechanisms; currently, it includes man-in-the-middle attacks and false data injection.

1 Introduction

Power systems are critical assets whose disruption can cause significant damage to the society investing all aspects, including human lives. Moreover the digitalisation of the power infrastructure allows the implementation of new functionalities (i.e. distributed energy resources, electric vehicle charging infrastructure, demand response, etc.) in order to better manage the power grid, but poses new challenges in terms of cybersecurity: the interaction of new actors by means of heterogeneous and third party networks widens the attack surface and every day new zero day vulnerabilities are discovered. Hence, it is crucial that any adversarial activity against the systems is detected as early as possible, enabling the adoption or strengthening of defense measures. Following the European Network and Information Security Directive 2016/1148, currently under review, the Member States have to ensure that essential entities, e.g. energy operators, take appropriate measures to manage the cybersecurity risks, including the adoption of methodologies to detect attacks and to actuate fast response and defense actions.

In this setting, we present a framework for the evaluation of cyberattack detection systems developed within a cooperation between researchers at the *Università del Piemonte Orientale* and at *Ricerca sul Sistema Energetico*, a company carrying out research activities in the electro-energy sector in national strategic projects of general public interest. Our research activity is aimed at the development of effective cyberattack detection tools for the energy systems, based on Artificial Intelligence methodologies. In a critical context such as the energy infrastructures, testing detection tools on fully operational infrastructures is unfeasible, and therefore a significant part of our activity is the design and implementation of a framework in which methodological results can be tested in a realistic setup.

In this report we refer to a scenario illustrated in Figure 1, considering Distributed Energy Resources (DERs). The power infrastructure evolution supported by new Information and Communication Technology (ICT) solutions allows the implementation of new functionalities

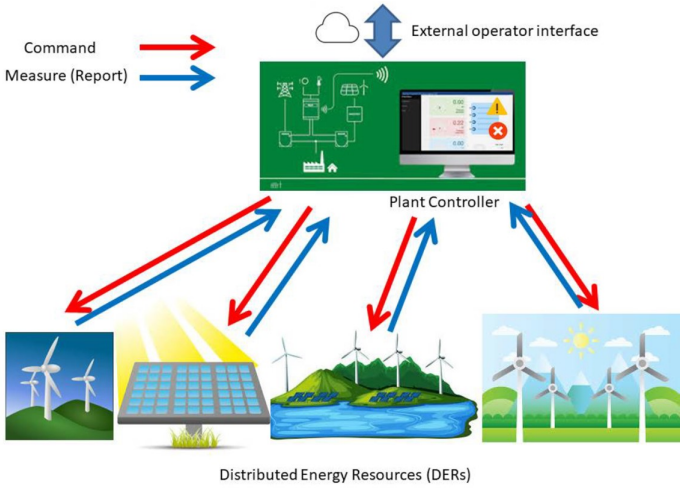

Figure 1: Use case.

with the involvement of new actors as for example the Significant Grid Users (SGUs). The increasing importance of the diffusion of the DERs requires to rethink the control logic including information coming from the field and controlling DER functions in terms, for instance, of reactive power. Communications need to be implemented from and to the energy resources (see Fig. 1) in order to collect measures and send commands. This information exchange can be implemented by means of heterogeneous and third party communication channels, hence they are prone to cyber security issues. National and international normative are working on standards to regulate the components and required communications.

In this report we first survey some related works, then we introduce the framework, explaining how attackers are emulated, giving an overview of the attacks that we implemented and describing the way we integrated detection models. Next the representation of the possible attacks using probabilistic graphical models (Dynamic Bayesian Networks - DBNs) is illustrated on a small but realistic example: the results that can be obtained by solving the DBNs are discussed and their possible use for monitoring the system is explained.

2 Related work

The idea of a testbed for Intrusion Detection Systems (IDS) in the environment of power control systems is not new, given the risk of conducting tests on a critical system.

The testbed structure proposed in [10] is quite articulated and various protocols are considered. But the attacks are injected in the system by altering traffic traces (`.pcap` files) and the goal is to test a machine learning based IDS. In contrast we propose a testbed in which an attacker is emulated, and therefore the malicious traffic produced is more realistic. Moreover the modularity of our approach enables the testing of different detection systems, to inject entire attack processes and to test early detection and forecast of adversarial actions.

In [8], a testbed for intrusion detection in an electricity generation and distribution control system is presented. In this case traffic is captured at the boundaries of a real Industrial Power System and comprises corporate and Operational Technology (OT) traffic. The testbed was used to verify the quality of a protection system [9]. IP traffic obtained from real control and supervisory substations was injected into the testbed domains. Hostile traffic was simulated as malformed traffic. This approach can detect traffic anomalies but is not geared at recognizing an attack process.

Another testbed for intrusion detection in the power grid is presented in [20]. In this case, the authors propose detection through a module that simulates the grid and compares measures reported with "ideal" data it computed. They simulate attacks by actually installing malware. Their focus is on single attacks and not on processes and the detection takes place when the attacker is already succeeding in destabilizing the grid.

An older approach is in [11]. Here the computer networks and the power grid are simulated. The detection tool is an anomaly detection tool, trained to recognize normal traffic in the net and to raise an alarm when the traffic is anomalous.

A much broader project is the US National SCADA Test Bed [15] to address cyber security issues of energy delivery systems. The project was started in 2003 and encompasses a number of subprojects such as quantum key distribution, risk assessment, removing certain classes of attackers, protocol compliance, development of control system standards.

The approach presented in [16] refers to an actual IDS (DETECT), rather than a testbed, however it is related with our proposal since Bayesian Networks derived from Attack Trees are used as models to recognize attacks as they evolve: events are gathered and monitored (within a given time frame), and observations are used to update the BN parameters. The model is solved after any event-driven update and may produce an attack warning or alarm. In our case Dynamic BNs derived from attack graphs are used in a similar way to detect emulated attacks.

For completeness, let us stress that testing IDSs is a problematic issue. Delving into this is out of the scope of this chapter, and we refer to a classical survey for an interesting analysis [12].

As will become clear in the following, our methodology and framework consist of coordinated components, such as the attack graph (see e.g. Figure 5), the selection of the attacks that the synthetic attacker will carry out (see Section 3.1) and the model (see Figure 8). In order to make the framework more flexible, allowing automated derivation of the three components of the attack/detection workflow, it would be useful to formalize attack scenarios by using some domain specific language. This formalization would allow us to derive in a coordinated way attack graphs of adversarial processes and, on the practical level, both the specifications to drive the emulated attacker and the corresponding detection models, thus supporting the development of more complex testbed scenarios. A good candidate domain specific language is that proposed in [6]: in this case the MAL language integrates the description of assets (servers, software, network components, ...) and of possible attack techniques (mainly derived from MITRE ATT&CK) that may apply to each asset, and possible coutermeasures. From the integration of such specification with the description of the assets composing the IT (or OT) infrastructure under study, attack graphs can be derived automatically and used to customize the testbed for experimenting with different scenarios.

3 The framework

Our framework is part of a testbed that reproduces a power control infrastructure where the focus is on communication implemented by standard protocols as IEC 61850 and IEC 60870-5-104. Figure 2 represents a diagram of the main control communications involved in the testbed:

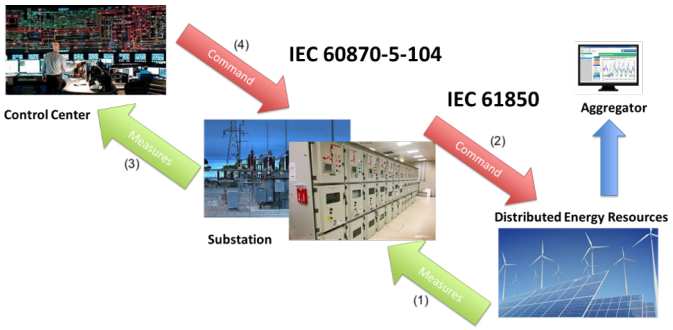

Figure 2: Testbed communications

the Distributed Energy Resources (DERs) send measures to a primary substation (arrow 1 in the figure) where control functions (e.g. voltage/frequency control) are executed and setpoints evaluated. These are then sent back to DERs (arrow 2 in the figure) in order to be applied. The communications are implemented by IEC 61850 standard.

The primary substation aggregates the measures coming from the DERs, integrates them with other power values and sends the data to a Control Center (arrow 3 in the figure). Here

higher level control functionalities are performed and commands could be forwarded to the primary substation (arrow 4 in the figure). The IEC 60870-5-104 protocol is used.

In this chapter the focus is on DER control (see Section 3.2): the communications are implemented by means of the MMS (Manufacturing Message Specification) protocol as defined in IEC 61850-8-1 standard. At the substation the MMS client is active and is connected with the MMS server placed at the DER site (see Fig. 3).

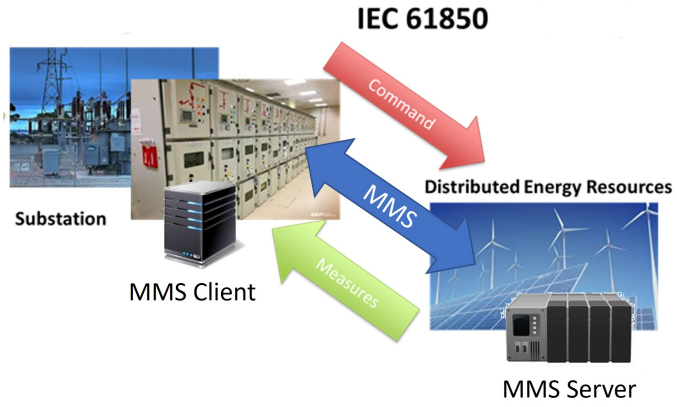

Figure 3: MMS communication

In this emulated environment, we inject adversarial activity, collect logs from the systems, analyze such logs and produce evidences that are later processed by models that can raise alerts, and give diagnostic or predictive information (see Fig. 4). The whole framework enables us to effectively test the adequacy of the detection mechanisms, comparing final results with the attacks we generated.

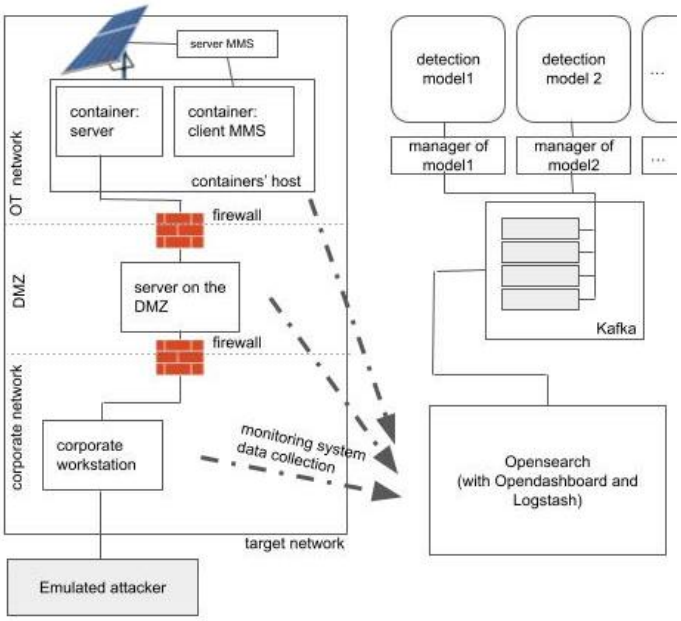

Figure 4: The logical architecture.

The adversary side is based on the tool Metasploit and it can emulate different behaviours, to render more aggressive or more naif attack attempts. It is capable of taking advantage of all the choices offered by Metasploit in terms of attacks, but we have also implemented more power-specific threats. Thus the malicious traffic is created in a realistic way.

The monitoring observes activity in the target network. Log data is conveyed to an Opensearch database which implements analytics, filtering the information it receives and raising alarms when specific events take place or defined thresholds are exceeded.

Such alarms are conveyed to the detection models through a Kafka broker, whose aim is to ensure modularity of the framework. It enables us to feed the same data to different models in order to compare their effectiveness for the different tasks of interest, such as early detection of adversarial activity, diagnosis of the attack steps carried out so far, prediction of most likely evolution of the attack processes.

The models we have so far integrated in the framework are probabilistic graphical models, in particular Bayesian Networks (BN), both static and dynamic (DBN) (see Section 4).

3.1 The attacker

In order to test the detection systems with complex attack processes, we implemented an automatic adversary. Our adversary is **fully automated** in the sense that it is capable of carrying out entire attack processes without the need of human intervention.

We wanted to mimick through our attacker **different adversarial profiles**, to test our detection tools in various settings, ranging from uninformed adversaries to very precise ones that know very well the network they attack.

We worked at making it **modular and expandable** in order to enrich over time the pool of attacks that can be carried out, with the specific aim of specializing increasingly over time our tool for power systems.

We opted for an **open source** tool, built over well established open source software, to provide a tool that can be of use to the community at large.

We could not use directly for our aims open source tools readily available because our aim is different from the usual one: attack tools are normally used for penetration testing. On the contrary, we know well the vulnerabilities of the network we emulate. Instead, we want to carry out those attacks in order to test our monitoring and detection tools. The environment in this sense is much more controlled: we know in advance what the attacker can achieve and this is key for validating the defense mechanisms.

We built our adversary around Metasploit, that has become the toolbox of our adversary. Some of the implementation choices we made, resulted in constraints on the emulated network, but we adhered to some basic principles in order to be able to retrieve realistic data through the monitoring system. These principles can be summarized as emulating and using actual software on victim machines, including the network stack, and carrying out real attacks on these machines.

3.2 Attacks

The design of our attacks and of the detection models is framed as much as possible in the MITRE ATT&CK classification [21], in order to maintain a common language shared in other research approaches and for adherence to real world scenarios, since the MITRE project itself is based on data from known attacks or adversarial software.

In this context, we also developed attacks specific for the power world, that implement techniques from the ICS ATT&CK matrix [22]. Moreover we implemented attack steps from the Containers ATT&CK matrix [23].

The full attack process chosen as a testbed considers an attacker that, having initially compromised a workstation in the corporate network, moves laterally through the DMZ to finally compromise a computer in the Operational Technology (OT) network, and from there proceeds to make the system unstable.

The final portion of the attack process we developed injects false data in the system: to this end, a Man in the Middle (MITM) attack to a IEC 61850/MMS over TLS connection is performed in the power control environment.

Our methodology consists in first designing attack processes that we express through *Attack Graphs* (AGs). Then on one side we implement the attacks for the synthetic attacker and on the other we implement monitoring sensors. The sensors must be realistic and not specifically targeted for the attack steps we selected: for instance we monitor the integrity of relevant

configuration files or directories, but we do not search for traffic patterns that expose one single exploit. On the basis of the AG, we also design the detection models.

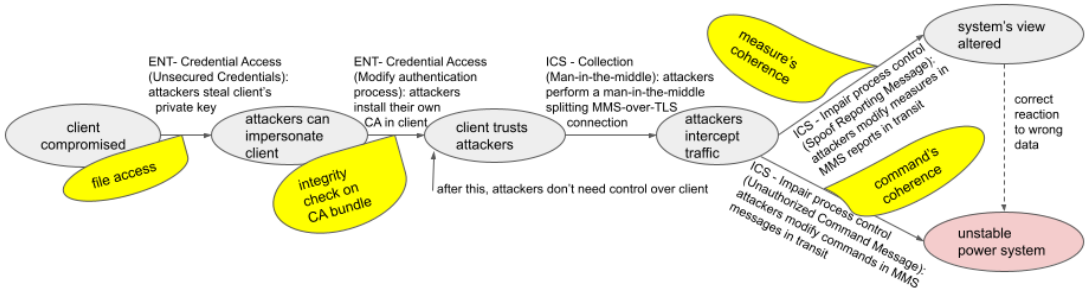

Figure 5: Man in the middle to MMS-over-TLS and related analytics

Figure 5 depicts an *Attack Graph (AG)* describing in detail the Man-in the-middle attack. Nodes denote states reached by the attacker and edges are labelled by attack steps. For each attack step we first indicate the MITRE ATT&CK tactic it aims at, in parentheses the specific technique, and then we detail the actual implementation of that technique. The tactic name is introduced by a code indicating the specific ATT&CK project we refer to; in Figure 5 some tactics are from the Enterprise ATT&CK matrix (ENT) and some are from the ICS one. So for instance, the edge labelled "*ICS - Collection (Man-in-the-middle): attackers perform a man-in-the-middle splitting MMS-over-TLS connection*" indicates an implementation of the technique Man-in-the-middle for the tactic Collection of the ICS matrix. In particular, the MITM attack is implemented splitting MMS over TLS.

The node labelled "unstable power system" is the attackers' goal that can be reached following two different attack paths: either the attackers modify reporting messages or commands. In the latter case, they can issue a wrong command that causes malfunctioning of the device driven by the MMS server. If on the other hand they alter reporting messages, the control center will have an incorrect view of the system's state. This might prompt reactions that can cause system's instability. A reaction to an altered view is not properly an attack step, and therefore it appears in the AG as a dotted arrow, but it is an action that the attackers count on to reach their target.

The yellow comments on some edges denote analytics to expose the attack. In this case four analytics are proposed. The one on the leftmost edge raises an alarm in case of access to the file containing the client's private key. Integrity check on the CA bundle (on the second edge from the left) is very important because any change to that file implies modified trust. The analytics on the rightmost edges are at application level: they check coherence over time of the measures received or of the commands issued.

The first portion of the whole process is a more classical lateral movement attack that proceeds scanning the network to identify possible victims and then exploits vulnerabilities of a mail server. Through these steps the attacker that has initially compromised a computer in the corporate network, most likely using standard phishing techniques, moves to the DMZ between corporate and OT networks. These attack steps are described in Figure 6. Notice that all techniques here are from the Enterprise ATT&CK matrix.

Perhaps more interesting is the attack to the containerization software; the subprocess is detailed in Figure 7: after compromising, with standard vulnerability exploitation, a container on a machine of the OT network, the attackers are able to escape from the container and take control of the whole host machine. Notably, they take control of all the containers running on the host, including the MMS client target of the attack in Figure 5. In Figure 7 most tactics are from the MITRE Container ATT&CK matrix (code CONT).

Whereas the emulated attacker is directed to pursue these specific attack paths, it is implemented to embed the attack processes that will be successful more realistically into a swarm of alternative attempts that are bound to fail, but create further challenges to diagnostic and predictive analyses.

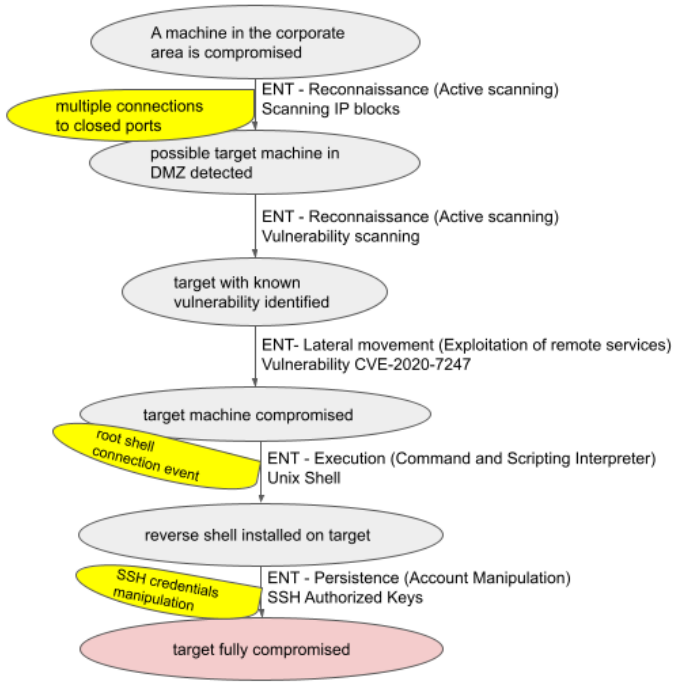

Figure 6: Lateral movement, with analytics

3.3 Integration of detection models

The monitoring system collects data from the network and conveys them to the Opensearch database. Here a pipeline filters the data and tests them against parameter thresholds or other specific conditions, configured according to the analytics that have been planned to expose possible attacks. When evidences of suspicious activity are detected, the alerts are forwarded to the detection models through a Kafka broker, on a specific topic.

Kafka consumers read the evidences and forward them to the detection models. As anticipated, we have so far integrated only (D)BN models, and so we describe here this integration. Our software includes a Kafka consumer that reads a Kafka topic at regular intervals (that map to the instants t in the DBN), collects them in a file according to a format that is compatible with Octave [5]. Then the program starts the DBN analysis: evidences are read from the file and one of the possible inference tasks are executed (see Section 4.2.1).

As the (D)BN manager receives new evidences, it starts a new analysis of the model.

The results produced by any model can be visualized in two ways. The most direct visualization is on a browser: an `html` file containing the data is produced, and this file can be viewed thanks to a Tomcat [7] server integrated in the framework. On the other hand, data are also redirected through Kafka back to Opensearch, where they can be visualized using Opendashboard. This option is more flexible, since one can implement different dashboards, depending on the specific purpose of the analysis.

4 Probabilistic graphical models

The detection models we have so far integrated in the framework are Bayesian Networks (BNs), which are defined by a directed acyclic graph where a node corresponds to a discrete random variable having a conditional dependence on its parent nodes. Dynamic Bayesian Networks (DBNs) extend BNs by providing an explicit discrete temporal dimension.

4.1 Bayesian Networks

Bayesian Networks (**BN**) (also known as *Belief Networks*) [17] are a widely used formalism for representing uncertain knowledge in Artificial Intelligence. They have then gained a great

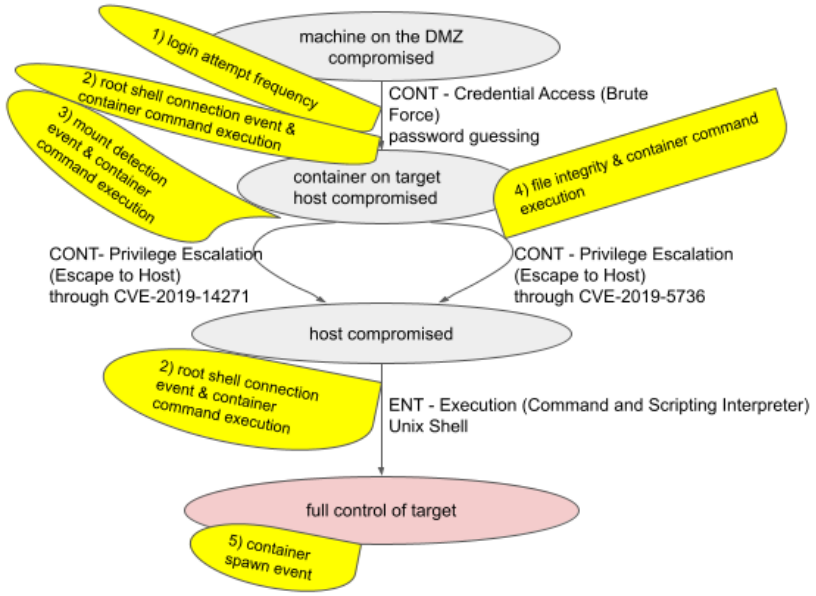

Figure 7: Attack to escape from a container, with analytics

popularity in the reliability field as well, because of their flexibility in modeling and analysing dependable systems [18].

A Bayesian Network (**BN**) is a pair $N = \langle\langle V, E\rangle, P\rangle$ where $\langle V, E\rangle$ are the nodes and the edges of a *Directed Acyclic Graph* (**DAG**) respectively, and P is a probability distribution over V. Discrete random variables $V = \{X_1, X_2, \ldots X_n\}$ are assigned to the nodes, while each edge $e \in E$ from node X to node Y represents a probabilistic relationship between the variables represented by X and Y, where Y directly depends on X. This interpretation allows us to factorize the joint probability of the variables of the model, by considering only the conditional distribution of each variable with respect to their parent variables in the DAG, as shown in equation 1.

$$P[X_1, X_2, \ldots, X_n] = \prod_{i=1}^{n} P[X_i | Parent(X_i)] \tag{1}$$

In the standard case of discrete variables, each local distribution can be described in tabular form (i.e., a column for each combination of states of the parent variables), called *Conditional Probability Table* (**CPT**).

Because of the availability of the joint probability distribution, any kind of probabilistic query of the form $P(Q|e)$ can be computed, where Q is any set of unobserved variables and e is a configuration of a set of observed variables called the *evidence*.

Although we have integrated also static BN models in our framework, we concentrate in the following on dynamic models, because they are more interesting, allowing more informative analyses, especially when used to approximate continuous models.

4.2 Dynamic Bayesian Networks

Given a set of time-dependent state variables $X_1, \ldots, X_n$, and given a BN N defined on such variables, a *Dynamic Bayesian Network* (**DBN**) [13, 19] is essentially a replication of N over two time slices $t - \Delta$ and t (Δ is the time discretization step), with the addition of a set of arcs representing the transition model. Let X_i^t denote the copy of variable X_i at time slice t, the transition model is defined through a distribution $P[X_i^t | X_i^{t-\Delta}, Y^{t-\Delta}, Y^t]$ where $Y^{t-\Delta}$ is any set of variables at slice $t - \Delta$ different from X_i (possibly the empty set), and Y^t is any set of variables at slice t different from X_i (possibly the empty set).

An edge connecting a variable $X_i^{t-\Delta}$ in the slice $t-\Delta$ to the variable X_j^t in the slice t, is called *temporal arc* (if $i = j$ the arc connects the two instances of the same variable). The dependency

of a given node on its parent nodes (possibly including its historical copy) is quantified in its CPT.

4.2.1 Inference tasks

DBNs feature several kinds of analyses [13], using different types of inference algorithms.

The inference is carried out by means of computation of posterior probabilities. Given a set of time stamped observations (called the *evidence*), the task is to determine the probability, at a given time point, of a set of variables of interest, given (i.e., conditioned on) the evidence. The following inference tasks can be performed:

- *Prediction* can support the analyst/system in identifying, on the basis of evidences obtained from the analytics, the subset of techniques that the attacker will more likely exploit in the future, thus enabling the setup of defensive actions. It consists in computing the probability of a future state, given past evidence. In other words, the prediction task consists in computing the posterior probability at time t of a set of queried variables Q, given a stream of observations $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t_1 < \ldots < t_k \leq t$. Every evidence e_{t_j} consists of a (possibly different) set of instantiated variables.
- A special case of prediction called *Filtering* occurs when the last evidence time point and the query time are the same ($t = t_k$). Filtering can support the security analyst or system in online diagnosis and early detection.
- *Smoothing* enables *diagnosis*, allowing determining the probable causes of security events. The task estimates a past state, given all the evidence up to now. So, the smoothing task consists in computing the probability at time t of a set of queried variables Q, given a stream of observations $e_{t_1}, \ldots, e_{t_k}$ from time t_1 to time t_k with $t < t_1 < \ldots < t_k$.

4.3 A DBN for the MITM attack

As an example to make the presentation more concrete, we show in Figure 8 a DBN that models (among others) the attack processes of Figure 5. This is only a portion of the full DBN that includes also the previous parts of the attack process, that is the lateral movement and the escape from the container, but the relevant concepts are clearly exemplified by this fragment. The central branches in the figure, starting from the low central node labelled "UnsecureCred", up to the top node, model the attack from Figure 5. The top node, "UnstablePS (OR)" represents the attack's goal: making the power system unstable. The branches to the right and to the left are different possible attack paths, that have not been implemented so far. We included such attacks, even though they currently cannot take place in our testbed, since we want to analyze how our model detects among others the attack process that is really taking place.

Other than the top node, all nodes represent either attack steps, e.g. edges in the AGs, or analytics, that in the AGs are denoted as yellow comments to edges (cf. Figure 5). The power system is unstable (the attacker's goal occurs), if any of the three last steps below it are successful. Analytics are leaves in the DBN graph: when an attack step takes place, traces are collected by sensors and alarms are raised by the analytic. Therefore in the DBN there is an arrow from a technique to an analytic, if the occurrence of the technique potentially causes an evidence from the analytic.

Nodes labelled "CorrReact" and "WrongLogicExec", near the top, are different from the rest. They represent actions that cause the attackers' plan to evolve but are not properly attack steps: "CorrReact" represents the reaction of the system automatic services to false measures injected by the attacker, corresponding to the dotted arrow in Figure 5 (see the discussion in Section 3.2). "WrongLogicExecution" is similar in concept: it is the final step of an attack process in which the attackers have modified the controller's logic, say with a malware, and therefore the system will react according to the malicious logic. Also in this latter case, the event is not properly an attack step but completes the process planned by the attackers. In both cases, we consider that some protection measures have been taken, in the form of some

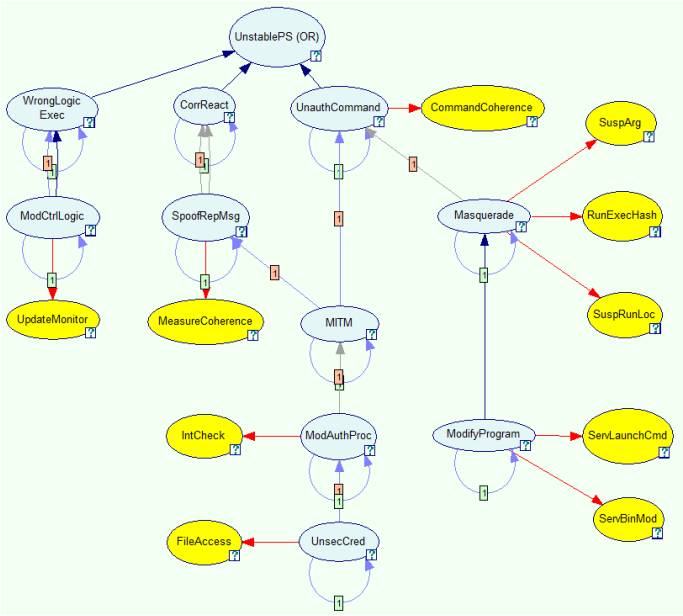

Figure 8: A portion of our Dynamic Bayesian Network

automatic control, and therefore these final steps don't take place necessarily but only with a certain probability.

Each attack step can occur with a certain probability, conditioned on the occurrence of its ancestors: so, for instance, the arrow from the node labelled "MITM" to the one labelled "SpoofRepMsg" implies that a reporting message can be spoofed only after a MITM has been successfully established. If a node has no ancestors, we are assuming that the prerequisites for that attack step are verified. Arcs from one node to itself model the fact that if an attack step has been carried out at time t, at time $t+1$ we acknowledge that it has already occurred. The *a priori probabilities* of nodes with no ancestors and the *conditional probabilities* of all other nodes are parameters that must be chosen with attention.

4.4 Representing attack paths timing

The DBNs are a discrete time model and as such it is capable to represent and evaluate probabilities that a certain attack step is carried out by attackers provided its preconditions are true, or compute the probabilities that a certain attack process takes place. But it cannot compute **probability distributions of completion times** of the different techniques or attack paths.

This kind of information would be very useful for security analysts, supporting them in decisions on how to react when under attack. For instance, assume that the analysts know that with a very high probability, say 90%, the attackers will not gain their goal before two days. This allows them to take advantage of this delay to carefully plan the defense, activating effective protection mechanisms and further detection tools, and increasing monitoring for a better post-mortem analysis. To the contrary, the information that the time left before the attackers reach their goals is less than two hours, with a probability of, say, 80%, prompts the analysts to rush to the protection of most important resources, even with drastic measures.

The above tasks require a continuous time model, which we approximated with our discrete time models.

Consider the behavior of the node "MITM" in the DBN of Figure 8: if node "ModAuthProc" has been activated, starting in state 0, at each time-step it has a constant probability p to jump at the next step in state 1, and $1-p$ to remain in state 0. In such a case the number of steps needed to get the first state transition is a geometric random variable X with parameter p. If we assume that each time-step has a constant duration Δt, the time required to make the transition can be computed as the sojourn time in state 0, which is equal to $k\Delta t$. When $\Delta t \to 0$ the time-step becomes an infinitesimal dt and the discrete geometric random variable corresponds in continuous time to an exponential random variable Y with rate λ, where the

probability to make the transition during an infinitesimal interval $[t, t+dt]$ is:

$$Pr(t \leq Y \leq t+dt) = \lambda dt. \tag{2}$$

Moreover, we know that in a geometric distribution the expected number of steps $E[X]$ of sojourning in state 0 is equal to $1/p$ and, in the analogous continuous case of the exponential distribution with rate λ, the expected sojourn time is:

$$E[Y] = \frac{1}{\lambda}. \tag{3}$$

Notice that according to Eq. 3, we can define an exponential random variable with a desired expected sojourn time $\overline{T}$ by setting the rate λ of its distribution to the inverse of $\overline{T}$.

Thus, to approximate the behavior of a single node with a continuous model such that its expected sojourn time is equal to a given value $\overline{T}$, we must build a discrete model with a proper choice of Δt and p. In particular, we must set:

$$\Delta t = \frac{\overline{T}}{m}; \qquad p = \frac{1}{\overline{T}}\Delta t = \frac{1}{\overline{T}}\frac{\overline{T}}{m} = \frac{1}{m}; \tag{4}$$

so Δt satisfies Eq. 3, to obtain the given sojourn time, and it is further divided by $m \in \mathbb{N}^+$ to achieve a desired approximation accuracy. The resulting probability p to make the jump during Δt must satisfy the discrete version of Eq. 2. In Table 1, we show the mean times to completion of each technique (and of all other event and status nodes), that is the desired sojourn time in state 0 of each node. One unit in Table 1 corresponds to 10 minutes.

Techniques, events, states		
Abbreviations	Full names	Average completion times
Masquerade	Masquerade	2
MITM	Man In The Middle	2
ModAuthProc	Modify Authentication Process	1
ModCtrlLogic	Modified Control Logic	50
ModifyProgram	Modify Program	2
SpoofRepMsg	Spoof Reporting Message	15
UnauthCommand	Unauthorized Command	40
UnsecCred	Unsecured Credentials	1
CorrReact	Correct Reaction	0
WrongLogicExec	Wrong Logic Execution	0
UnstablePS	Unstable Power System	0

Table 1: Estimated completion times of each technique in Figure 8

The same approach can be applied to all technique-nodes in the DBN, however care is needed to cope with concurrent activities. For instance accordingly to their CPTs, when "MITM" becomes active (state 1), the nodes "SpoofRepMsg" and "UnauthCommand" represent two concurrent activities all enabled at the same moment and racing against each other to jump in state 1; the first winning the race will make the jump. The time at which such jump will occur still follows an exponential distribution with a rate equal to the sum of the rates of each concurrent activities. For such reasons, to apply the approximation to the whole network, Eq. 4 must be extended to:

$$\Delta t = \frac{1}{m\sum_i (1/\overline{T_i})}; \qquad p_i = \frac{\Delta t}{\overline{T_i}}, \tag{5}$$

where $\overline{T_i}$ and p_i are the desired sojourn times and resulting jump probabilities of each technique, respectively. The introduction of the sum of the inverse of the sojourn times is required to model concurrency. More details can be found in the explanation of the *uniformization method* [14].

In this way we obtain a discrete-time DBN in which all conditional probabilities are re-scaled with respect to the computed ΔT. The approximation of the continuous-time model is obtained assuming that a time sufficiently small with respect to the specific parameters in input passes between two steps of the DBN with re-scaled parameters. This approximation allows to recompute distribution probabilities of the *time to compromise* (TTC) of the different

Table 2: Conditional Probability Table of the variable *ModAuthProc*

entry	*UnsecCred*	*ModAuthProc* (t-1)	*ModAuthProc* (t)	probability
1	0	0	0	1
2	0	0	1	0
3	0	1	0	1
4	0	1	1	0
5	1	0	0	0.931
6	1	0	1	0.069
7	1	1	0	0
8	1	1	1	1

techniques and attack paths, and by construction ensures that the mean times of the single techniques are the same as those given as input parameters. Moreover also the mean time of the TTC of an attack sequence is equal to the sum of the mean values of all the attack steps of the sequence.

In the the DBN (Figure 8) "UnsecCred" is the parent node of "ModAuthProc"; in other words, "UnsecCred" influences "ModAuthProc" because the activation of the first technique allows the second one to be attempted. Besides depending on "UnsecCred", in the Conditional Probability Table (CPT) (Table 2) "ModAuthProc" at time t depends on its copy at time $t-1$, in order to represent its discrete temporal evolution. In the CPT the activation of a technique is represented by the value 1 (0 otherwise). Each entry of the CPT represents a specific state configuration of "UnsecCred" and "ModAuthProc" (at time $t-1$), and provides the probability that "ModAuthProc" (at time t) evolves towards the state 0 or 1. In particular, in the entries 1-4 we have the situation where "UnsecCred" has not happened yet, so the probability that "ModAuthProc" occurs (state 1) is null. In the entries 5-6 "UnsecCred" has occurred and "ModAuthProc" is not active yet at time $t-1$; therefore "ModAuthProc" may occur (1) or not (0) at time t with a probability established according to equation 4 and the completion time defined in Table 1. Finally, in the entries 7-8, "ModAuthProc" is already active (1) at time $t-1$, and due to the absence of countermeasures in the model, "ModAuthProc" will maintain its state at time t.

5 Experiments

The experiments carried out involve several aspects. We have started with functional experiments showing that the integration of all components works as expected. The attacker carries out the prescribed attack processes, sensors collect data and send it to the database, the Opensearch pipeline forwards the data to the model managers through Kafka and the models are activated. In the following we report how variations in the behaviour of the attacker has led to different inferences of the model.

Recall that the attacker can carry out the attack processes in the center of Figure 8. The "File access" analytic monitors that there are no suspicious accesses to the private key file; it is implemented using the Linux Audit Framework [2] together with the `auditd` module of Auditbeat [4]. "Integrity check" raises an alert if the CA bundle has been altered; this is implemented using the `file_integrity` module of Auditbeat. The two coherence analytics were implemented collecting syslog data and analyzing it in Opensearch. We also implemented the "ServLaunchCmd" analytic, even if it is related to an attack process that we don't support yet, to monitor additional activity: recall that our synthetic attacker embeds the attack processes in additional, randomly selected attack attempts that are generally bound not to succeed. This analytic is implemented again using Auditbeat, and it looks for occurrences of the `Systemctl` command.

In our experiments, all the analytics that we haven't implemented yet will give no evidences. We stress that "no evidence" from an analytic is different from negative evidence. In the former case, we have not implemented the analytics and therefore the detection system has no information about possible adversarial activity in that domain. In the latter, the sensors are

deployed but detect nothing suspicious. In the model we assume that the analytics are imperfect but still very trustworthy: a technique may go undetected (false negative), or an alarm may be raised even without the attempt to use the technique (false positive). In both cases we assume a high confidence in the monitoring system and set the error probability to 10^{-4}.

In this paper, to design and evaluate the DBN, we use the *Bayes Net Toolbox for Matlab* (BNT) [1], and in particular, we resort to the Boyen-Koller (BK) inference algorithm [3]. BK is a general inference algorithm for DBNs that can perform both exact and approximate inference by trading-off precision (accuracy) of the posterior probability estimation and computational (time and space) resources. In the present work we experimented with the version of BK able to perform exact inference on the DBN model.

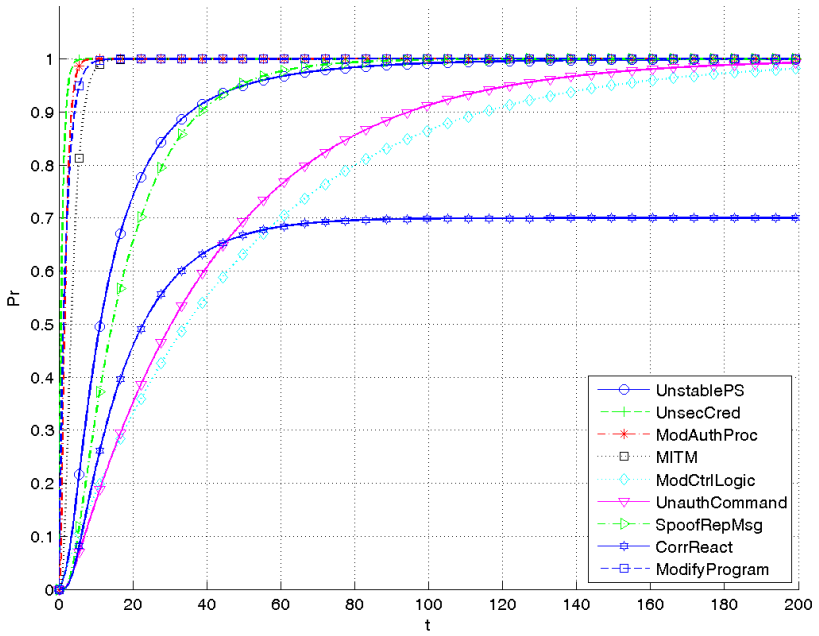

Figure 9: Probabilities of successful technique exploitation with no evidences.

5.1 No monitoring

In the first experiment we compute the likelihood that the techniques are successfully exploited by the attacker, in the setting in which an attack is carried out, but the monitoring system is not deployed into the architecture, thus no evidences are collected. For this experiment, we turned off the implemented monitoring system. Notice that in this case, since no monitoring has been deployed, it is irrelevant whether the synthetic attacker is at work or not, since there's no monitoring system to record its activity anyway. Results in this setting provide useful insights about the structural characteristics of the attack process according to the apriori probabilities derived from the estimated completion times. Fig. 9 shows the cumulative probabilities of success of the different techniques. According to the attack process modeled in the DBN in Fig. 8, after compromising a machine in the OT network, the attacker can modify the DER control logic (labeled "ModCtrlLogic"), obtain insecurely stored credentials ("UnsecCred") or modify a program of the plant controller ("ModifyProgram"). The difference between the success probabilities of such techniques depends on their average completion times, thus as expected it is more likely that the attacker completes "UnsecCred" attack since it has the shortest average completion time, then it can be followed by "ModifyProgram' or the sequence of "ModAuthProc" and "MITM" techniques. The "ModifyProgram" attack is less likely because it takes a significant higher time to be completed. Generally, we can observe an inverse correlation between the success probability of a technique at a given time and its distance from the final goal since the chance of success of a high-level attack step depends on the success of lower steps in

the attack path and it is further modulated according to its average completion time. Moreover, we can notice that in the long run all considered techniques will eventually succeed, except for the final step "CorrReact". As explained in Sec. 4.3 such step depends on the activation of a protection measure and thus it can be successful only with a given probability, fixed to 0.7 in all the experiments.

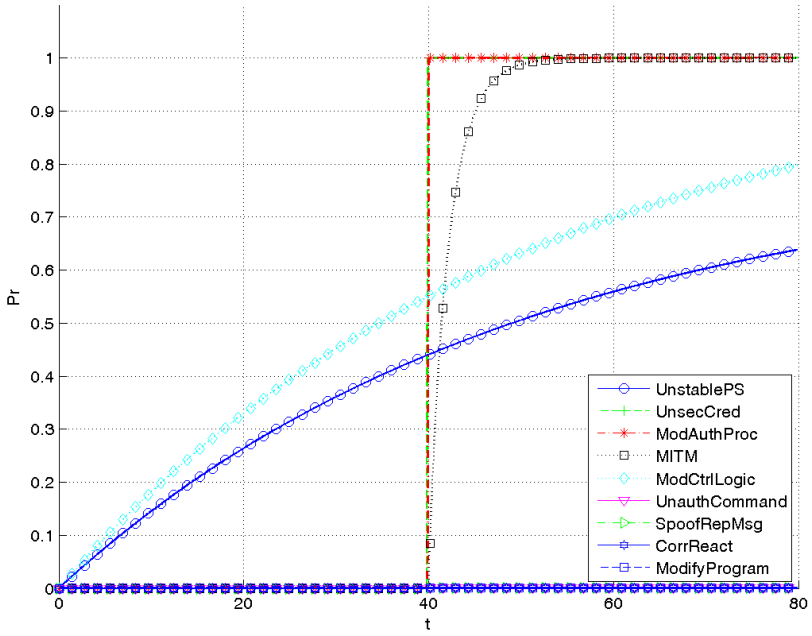

Figure 10: Probabilities of successful technique exploitation when the attacker performs attack steps "UnsecCred" and "ModAuthProc" just before $t = 40$ (analytics "FileAccess" and "IntCheck" up from $t = 40$, all other implemented analytics down, the remaining undefined (no evidence).)

5.2 Two attack steps

Next, we turned on the monitoring system and observed how the collected evidences affected the results. We started the adversary with a delay and had it carry out the first two attack steps in rapid sequence; evidences of its activity reached the DBN at time $t = 40$. The model assumes that an attack is underway starting from $t = 0$, which models a typical safe posture of the security analyst: the attacker is never inactive, as far as we know. But, until $t = 40$, all the implemented analytics certify there is no adversarial activity; at that point both "FileAccess" and "IntegrityCheck" raise an alert. The alerts, once raised, are never cleared in the DBN, and so the evidences are considered positive until the end of the analysis at $t = 80$ (see Section 6). All other implemented analytics are still giving no alarm. We can observe in Fig. 10 that initially, without any positive or negative evidences from the "UpdateMonitor" analytic, and negative evidence from all the others, the DBN model infers that the attacker can only attempt a "ModCtrlLogic" attack; then "WrongLogicExecution" might occur and thus there is a probability, increasing over time, that the system be compromised through attacks that cannot be detected. When both "FileAccess" and "IntegrityCheck" start to raise alerts, the model correctly infers that the attacker is taking the central right attack path that goes from "UnsecCred" to "ModAutProc" and that the following more likely step will be the exploitation of a Man-In-The-Middle attack splitting MMS over the TLS connection. Note that, even if such step is successful, the model derives that the attacker is not able to go further along this path, since both analytics "MeasureCoherence" and "UnauthCommand" are not raising alerts. Therefore the evidences triggered by the first attack steps carried out by our synthetic attacker have no impact on the forecasts of system instability.

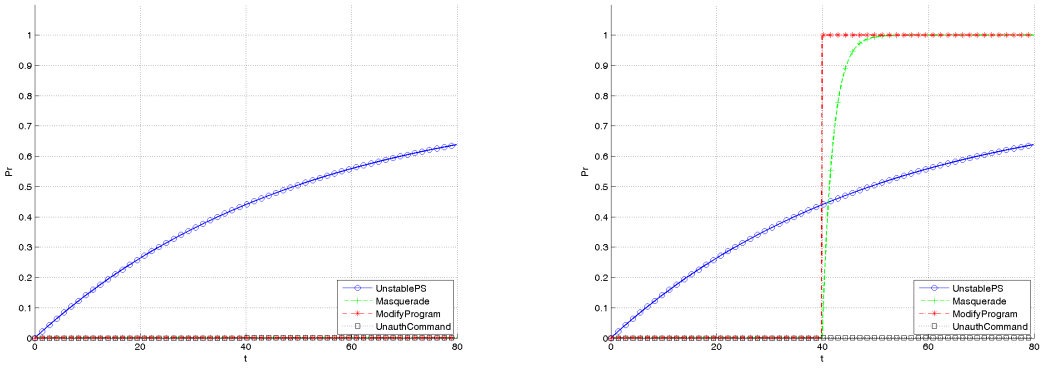

Figure 11: Probabilities of successful technique exploitation of the techniques in the attack path starting with "ModifyProgram". At left: in the same setting of Figure 10; at right: when the attacker also tries to stop a service triggering analytic "ServLaunchCmd".

5.3 Disturbances

Recall that our attacker can add randomly selected attacks to the attack paths, to mimic a real attacker trying different possibilities, and create disturbances in the detection system. In the following experiment we replicated the activity of Section 5.2, but the attacker also tried to stop a Tomcat server (not running) in the machine running the SCADA system. This attempt triggered an alert of the "ServLaunchCmd"analytic. To highlight the differences, we plot in Figure 11 only the probabilities of the techniques in the rightmost attack path of Figure 8. On the left side, we show the probabilities in the setting of the experiment of Section 5.2. Here we see that all techniques are considered not to occur because the only analytic deployed for the first step ("ModifyProgram") is raising no alarm: all other attack steps can take place only after this one. The growing probability of an unstable system is due to the reasons discussed in Section 5.2.

On the right side, we show what the DBN model infers when our attacker is also trying to stop the service. This time, adversarial activity related with the "ModifyProgram" technique is detected. Thus, the detection system expects with high probability that "ModifyProgram" has been carried out. Since no monitoring for a "Masquerade" technique is deployed, the DBN can only warn that with high probability also that attack step will be soon carried out. But, we have an analytic in place to detect occurrence of an unauthorized command, and this is raising no alarm. Then even though there is suspicion that the attacker has started along this attack path, the analyst can be sure that no further steps have been taken so far and therefore no impact on the instability of the system is expected (UnauthCommand has zero probability to occur and the curve for UnstablePS is exactly as on the lefthand side).

5.4 One whole attack process

Here we had our attacker carry out one complete attack process: from "UnsecCred" all the way through "SpoofRepMsg". The monitoring system detects that the measures sent to the server all of a sudden are not coherent, and thus also "MeasureCoherence" signals a suspicious activity at $t = 40$. The conclusions of the model are shown in Fig. 12. In this case we can derive that the attacker is also able to modify reporting messages. We assumed that in this case with probability 0.7 the automatic protections of the control system are not able to detect that something is wrong with the reported measures and to disable a reaction, thus causing the power system instability. Notice in Figure 12 how the probability of an unstable system increases when "MeasureCoherence" raises an alert.

This experiment highlighted one interesting aspect that requires further discussion: we had estimated (Table 1) a rather long execution time for the attack steps that injects false measures ("SpoofRepMsg"). Our attacker has altered dramatically the measures and thus the attack step was detected right away. On the one hand, this is unrealistic and says that we should parametrize the attacker more carefully. On the other hand this shows how in our model evidences take

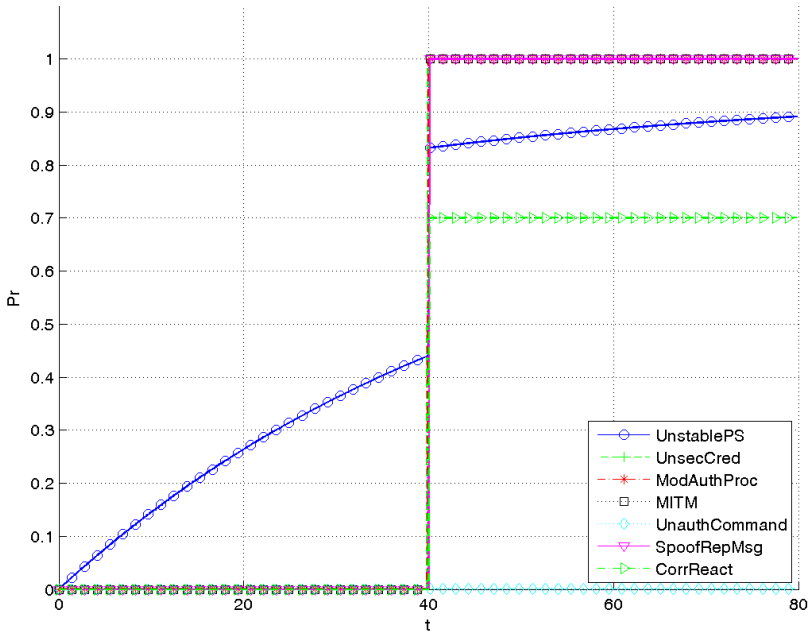

Figure 12: Probabilities of successful technique exploitation in case of a full attack process (analytics "FileAccess", "IntCheck", "MeasureCoherence" up from $t = 40$, other implemented analytics down, the remaining analytics undefined (no evidence).)

priority over any other consideration. We discuss evidences further in Section 6.

5.5 A slower attack

The model can also cope with a sequence of observations spread over time. In the experiment shown in Fig. 13 we had the attacker wait some random time between attack steps. So the alert from analytic "FileAccess" has been raised at $t = 6$, then at $t = 9$ "IntCheck" raised an alert, and finally "MeasureCoherence" at $t = 35$. Again all other implemented analytics remained inactive for all the time. The DBN infers the same probabilities for the techniques we are interested in, as in Figure 12. In Figure 13, though, since alerts are raised at different time-instants, it can be seen more clearly what the DBN infers from each alert. In particular notice that the occurrence of a MITM starts to be expected only after it is reported that the authentication process has been modified ("ModAuthProc"). Also notice that only when the analytic for "SpoofRepMsg" becomes active, then the model infers a higher probability of reaction to the injected measures, and as a consequence also the probability of an unstable system increases abruptly.

5.6 Slow and complete attack

This time, we had the attacker carry out both the attack processes that have been implemented. Again we timed its activity randomly. We report the probabilities inferred by the DBN in Figure 14. The times at which alerts were reported have changed with respect to Figure 13 since the adversarial timings are randomized in the current experiment and the previous. It is interesting to notice here, is that after the attacker injects a command ("UnauthCommand", detected at $t = 52$), the probability that the system will be unstable jumps to 1.

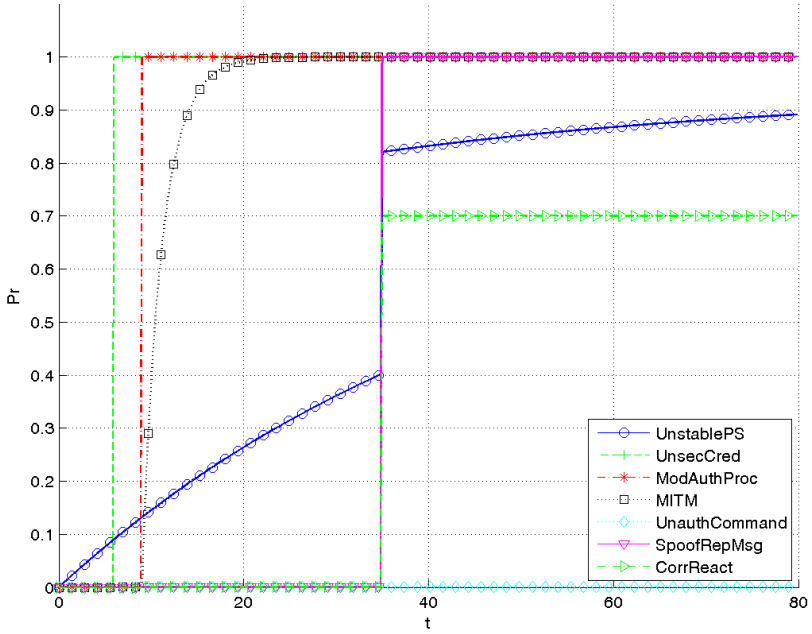

Figure 13: Probabilities of successful technique exploitation with a slower attack (analytics "FileAccess", "IntCheck", "MeasureCoherence" activated at $t = 6$, $t = 9$ and $t = 35$, respectively; other implemented analytics down, the remaining analytics undefined (no evidence).)

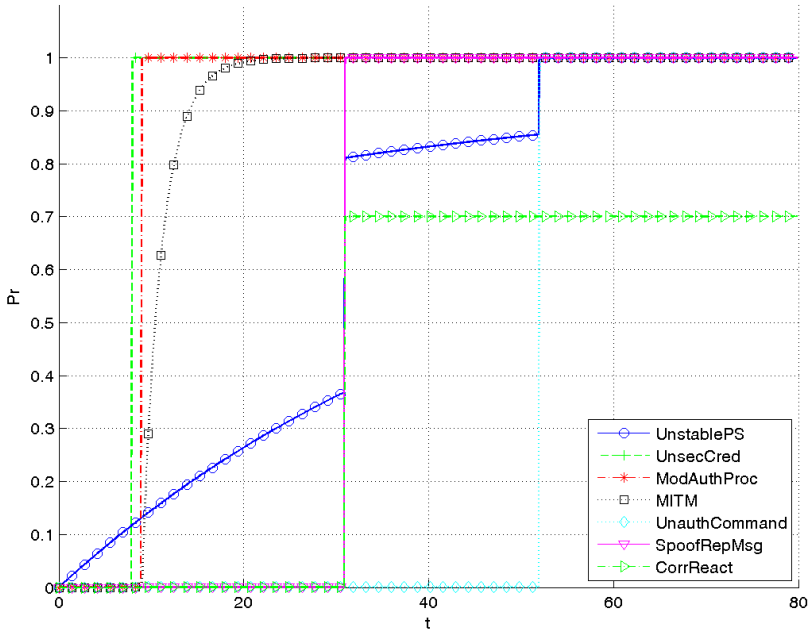

Figure 14: Probabilities of successful technique exploitation when the attacker carries out both attack processes with randomized timing (analytics "FileAccess", "IntCheck", "MeasureCoherence" and "CommandCoherence" activated at $t = 8$, $t = 9$, $t = 31$, $t = 52$ respectively; other implemented analytics down, the remaining analytics undefined (no evidence).)

6 Conclusions and future directions

We have implemented a test-bed for the evaluation of detection models in which we emulate an attacker and deploy a monitoring system. The framework is modular in that it allows testing different detection mechanisms, possibly in parallel, but also enables defining a specific profile for the attacker, and adding or removing attack steps.

In the first testing phases we have verified the interoperability of the various components and that the data flow is as expected. We have seen the models mirror the activity of the attacker. On the other hand, the experiments have exposed some shortcomings of the models: our current models don't capture the different attacker profiles, the analytics remain valid indefinitely. See below for the discussion of these issues, but bear in mind that the goal of the framework is precisely exposing possible inadequacies of the models we test.

Our DBNs are so far parametrized only based on estimated completion times of the attacks. This means that there is no attempt, as yet, at modelling the attacker's profile. The profile is important when the detection model must integrate incomplete information from the monitoring system. We plan to introduce features to model different attacker profiles in the DBN, matching the various behaviours that can be configured for the synthetic attacker. In order to parametrize correctly the attacker's behaviour and the DBN that models it, it is necessary to have access to extensive data on real world attacks. The MITRE ATT&CK project is a step in that direction, but unfortunately the knowledge of the real scenario is currently limited, partly because not all attacks are detected, partly because not all of those detected are reported. These limits are even more evident in the ICS ATT&CK project. Moreover, that project considers all industrial and critical infrastructures, whereas we would like to have data specific for the power infrastructure. Therefore, currently, we cannot count on using realistic data, as a consequence in the test process we also intend to determine the robustness of the models, analyzing how much the results are affected by a discrepancy between the actual adversarial behaviour and the assumptions expressed by BNs' parameters.

On the other hand, a more realistic emulation of the attacker would allow us to create realistic traffic in the network and gather traces that could in turn be used to train machine learning models, to increase our detection modules.

After these preliminary tests, our aim is to use the framework to understand the potentialities of the models we integrate in it, in particular of the DBNs. In the experiments we presented here we only used filtering for the analyses, that is we concentrated on online diagnosis and early detection. Through our framework's dashboard the analysts will see graphs like those in Figures 10 through Figures 14 limited to the time instant of the observation; for instance all of our figures assume that the analyst is watching the dashboard at $t = 80$. With DBNs we can carry out also predictive analyses for times beyond the current instant t; or diagnosis that tries to determine (past) causes of events detected by the monitoring system. Within our framework, besides testing the validity of the model results, we also intend to study an effective way to present all of this information in a way that supports the evaluation of the security posture.

Currently, when an alert is raised from an analytic in our DBN model, it remains active for the rest of the analysis. We plan on introducing an ageing mechanism for analytics, in the spirit of [16]. This would enable to discard an alert as a false alarm if it is not followed by additional indications of suspicious activity in due time. The ageing parameter is again a delicate issue, since often the attacker chooses to act slowly precisely to avoid detection.

Detection models offers added value with respect to the raw observation of the alerts from the monitoring system, allowing to derive more articulate information than just the occurrence of observable events. It is key understanding how to exploit this possibility in the most effective way. Our framework is a valuable tool to help analyze and unleash these potentialities.

Acknowledgments

This work is original and has been supported by a joint collaboration between RSE S.p.A. and Università del Piemonte Orientale, partially funded by the Research Fund for the Italian

Electrical System in compliance with the Decree of Minister of Economic Development April 16, 2018.

References

[1] Bayes Net Toolbox for Matlab. https://github.com/bayesnet/bnt.

[2] Linux audit, Accessed April 2022. https://linux-audit.com/.

[3] Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence—UAI 1998*, pages 33–42. San Francisco: Morgan Kaufmann, 1998. Available at http://www.cs.stanford.edu/~xb/uai98/.

[4] Elasticsearch B.V. Auditbeat, Accessed April 2022. https://www.elastic.co/beats/auditbeat.

[5] John W. Eaton. Octave, Accessed April 2022. https://www.gnu.org/software/octave/.

[6] Mathias Ekstedt. Meta attack language (mal). https://foreseeti.com/meta-attack-language/, Accessed April 2022.

[7] The Apache Software Foundation. Apache Tomcat. https://tomcat.apache.org/, Accessed December 2021.

[8] Jacek Jarmakiewicz, Krzysztof Maślanka, and Krzysztof Parobczak. Development of cyber security testbed for critical infrastructure. In *2015 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–10. IEEE, 2015.

[9] Jacek Jarmakiewicz, Krzysztof Parobczak, and Krzysztof Maślanka. Cybersecurity protection for power grid control infrastructures. *International Journal of Critical Infrastructure Protection*, 18:20–33, 2017.

[10] Seokcheol Lee, Seokjun Lee, Hyunguk Yoo, Sungmoon Kwon, and Taeshik Shon. Design and implementation of cybersecurity testbed for industrial IoT systems. *J. Supercomput*, 74:4506–4520, 2018.

[11] M. Mallouhi, Y. Al-Nashif, D. Cox, T. Chadaga, and S. Hariri. A testbed for analyzing security of SCADA control systems (TASSCS). In *ISGT 2011*, pages 1–7. IEEE, 2011.

[12] Peter Mell, Vincent Hu, Richard Lippmann, Josh Haines, and Marc Zissman. An overview of issues in testing intrusion detection systems. 2003.

[13] K.P. Murphy. Dynamic bayesian networks: representation, inference and learning. Technical report, University of California, Berkeley, 2002.

[14] V. Nicola, P. Heidelberger, and P. Shahabuddin. Uniformization and exponential transformation: Techniques for fast simulation of highly dependable non-markovian systems. In *1992 FTCS - The Twenty-Second Int. Symp. on Fault-Tolerant Computing*, USA, jul 1992. IEEE Computer Society.

[15] U.S. Department of Energy. National scada test bed, 2003-2022. www.energy.gov/oe/technology-development/energy-delivery-systems-cybersecurity/national-scada-test-bed.

[16] Mauro José Pappaterra and Francesco Flammini. *Bayesian Networks for Online Cybersecurity Threat Detection*, pages 129–159. Springer International Publishing, Cham, 2021.

[17] J. Pearl. *Probabilistic reasining in intelligent systems: networks of plausible inference*. Morgan Kaufmann, USA, 1988.

[18] L. Portinale and D. Codetta. *Modeling and analysis of dependable systems: a probabilistic graphical model perspective*. World Sc., USA, 2015.

[19] L. Portinale, D. Codetta-Raiteri, and S. Montani. Supporting reliability engineers in exploiting the power of dynamic bayesian networks. *International Journal of Approximate Reasoning*, 51(2):179–195, 2010.

[20] Prateek Singh, Saurabh Garg, Vinod Kumar, and Zia Saquib. A testbed for scada cyber security and intrusion detection. In *2015 Int. Conf. on Cyber Security of Smart Cities, Industrial Control System and Communications (SSIC)*, pages 1–6. IEEE, 2015.

[21] The MITRE Corporation. Adversarial tactics, techniques and common knowledge (ATT&CK), 2015. `https://attack.mitre.org/`.

[22] The MITRE Corporation. ATT&CK for industrial control systems, 2020. `https://collaborate.mitre.org/attackics/index.php/Main_Page`.

[23] The MITRE Corporation. ATT&CK for containers, 2021. `https://collaborate.mitre.org/matrices/enterprise/containers/`.