applied sciences

IMPACT FACTOR 2.5

CITESCORE 5.5

Article

Evaluation of Cyberattack Detection Models in Power Grids: Automated Generation of Attack Processes

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

Special Issue

Advanced Smart Grid Technologies, Applications and Challenges

Edited by

Dr. Paulo Tavares and Dr. Álvaro Gomes

MDPI

https://doi.org/10.3390/app151910677

applied sciences

MDPI

Article

Evaluation of Cyberattack Detection Models in Power Grids: Automated Generation of Attack Processes

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

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

Abstract

The recent growing adversarial activity against critical systems, such as the power grid, has raised attention on the necessity of appropriate measures to manage the related risks. In this setting, our research focuses on developing tools for early detection of adversarial activities, taking into account the specificities of the energy sector. We developed a framework to design and deploy AI-based detection models, and since one cannot risk disrupting regular operation with on-site tests, we also included a testbed for evaluation and fine-tuning. In the test environment, adversarial activity that produces realistic artifacts can be injected and monitored, and evidence analyzed by the detection models. In this paper we concentrate on the emulation of attacks inside our framework: A tool called SecuriDN is used to define, through a graphical interface, the network in terms of devices, applications, and protection mechanisms. Using this information, SecuriDN produces sequences of attack steps (based on the MITRE ATT&CK project) that are interpreted and executed by software called Netsploit. A case study related to Distributed Energy Resources is presented in order to show the process stages, highlight the possibilities given by our framework, and discuss possible limitations and future improvements.

Keywords: power grids; cybersecurity; network modeling; attack graphs; attack emulation; MITRE ATT&CK framework; IEC 61850

Academic Editors: Paulo Tavares and Álvaro Gomes

Received: 11 August 2025
Revised: 19 September 2025
Accepted: 23 September 2025
Published: 2 October 2025

Citation: Cerotti, D.; Codetta Raiteri, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Savarro, D.; Terruggia, R. Evaluation of Cyberattack Detection Models in Power Grids: Automated Generation of Attack Processes. *Appl. Sci.* **2025**, *15*, 10677. https://doi.org/10.3390/app151910677

1. Introduction

The complexity of modern power grids and their critical role in our society call for particular attention in the domain of cybersecurity. We have witnessed in recent years several significant attacks such as Stuxnet [1], BlackEnergy [2], and Industroyer 1 and 2 [3]. European directives also target the issue and recommend taking all necessary measures to limit the risks [4].

In this context, in a cooperation between the University of Eastern Piemont (UPO) and Ricerca Sistema Energetico S.p.A. (RSE), we have developed a framework for the design and deployment of AI-based detection models and a testbed for their evaluation and fine-tuning. Our main goals in the definition of the system are user-friendliness, limited necessary competences, and adherence to the MITRE ATT&CK classification. We described

in [5] the aspects related to detection, whereas in this paper we concentrate on the emulation of attacks.

It is obviously dangerous validating security functions for critical systems in place; apart from the damage that could result from discovering that the functions are ineffective in front of real-world attacks, it is inappropriate and risky to emulate attacks that could impair the functions of the system. So it is customary to create a test environment in which realistic but fully controlled adversarial activity is injected. A monitoring system on the emulated network collects the data and conveys it to detection models whose reactions are thus observed and evaluated.

1.1. Contributions

Our proposal for the emulation of the attack processes follows the same criteria as for the detection models: Through a graphical interface, and in a framework that is based on the MITRE ATT&CK project, definitions of the network, devices, applications, and protection mechanisms and articulated sequences of attack steps are produced. These sequences are interpreted and executed by a software engine. The attacks are actually carried out, and so the artifacts left on the victim machines and in the network by the process are realistic.

We implemented a graphical tool called SecuriDN, which is based on the framework DrawNET Modeling System (see Section 3.2). The software for emulation of attacks is called Netsploit (see Section 6), because it has Metasploit at its core, although it can also carry out offensive steps that are not available in Metasploit.

The novelty in our proposal lies in the integration of our emulation tool within a framework specifically designed to facilitate the setup of automated experiments by security experts, which includes the following:

- A graphical interface enables security experts to reproduce the system configuration;
- Vulnerability information is derived from the MITRE ATT&CK projects;
- Attack processes can be derived automatically;
- A synthetic attacker executes the attacks in an emulated network, producing realistic artifacts of adversarial activity;
- All of this is complemented with a monitoring system and detection models (not the focus of this paper, but briefly presented in an overview in Section 4).

1.2. Structure of the Paper

In the next section we discuss related approaches. In Section 3 we present the system background. Then, Section 4 presents in more detail the platform that we have sketched above. Section 5 describes how the attack sequence is generated starting from the complete network specification. In Section 6 further details regarding the synthetic attacker are presented. In Section 7 a SecuriDN use case is presented starting from the network specification up to the attack sequence generation. Conclusions, possible limitations and future developments are discussed in Section 9.

2. Related Work

There is extensive literature on platforms whose aim is either cybersecurity posture evaluation, cybersecurity education or software security testing. In our discussion of related work, we focus on approaches that are close to ours because they target critical systems, in particular the smart grid, and whose aim is cybersecurity posture evaluation. In the final part of this section we specifically address tools for attack emulations, given their centrality in our research.

Carrying out tests on security measures in critical systems is very risky since they might cause unexpected reactions that make the system unavailable, possibly damage costly assets, disrupt services, and that could even result in loss of lives. Therefore, we are certainly not the first to use an emulated environment as a testbed for Intrusion Detection Systems (IDSs). Here we are not interested in surveying the testbeds in their entirety, but rather in looking at how the adversarial behavior is injected into them. See for instance [6] for a broader analysis of the literature.

An interesting survey is [7], from 2020. Among the platforms reviewed in [7], the only two that are emulation-based and whose aim is security software testing are [8,9]. The former specifically targets power grids. In both cases, we found no description of automated injection of adversarial activity.

A more recent (2021) survey on testbeds for ICS systems [10] shows that many testbeds specific to power grids are either not meant for cybersecurity research or are not available to the public. We find there a proposal close to our interests [11], from 2019: They specifically implement an IEC 61850 standard, including the MMS communication protocol (see Section 6). The paper focuses on the performance of the testbed without discussing the features and results relative to attacks; there is no description of how attacks are injected, whether attack processes are automated, or which results they achieve.

Another recent approach is [12] (2022). They implement a minimal power system testbed with a voltage transformer process, an HMI, a SCADA system, and a database. Among other protocols, they offer IEC 61850 over MMS. They implemented four MITRE ATT&CK techniques (see Section 3.1) and they provide `.pcap` traces of regular traffic and cyber attacks. The paper does not describe how attacks are orchestrated.

In the testbed structure proposed in [13], the attacks are injected in the system by altering traffic traces (`.pcap` files). In contrast, we propose to emulate the attacker to produce more realistic traffic patterns, and, given the flexibility of our approach, we can easily fine-tune our attacker to produce traffic traces with various characteristics.

A testbed for intrusion detection in an electricity generation and distribution control system is described in [14]. Here the adversarial traffic is simulated as malformed traffic, starting from real traffic captured at the boundaries of a real industrial power system. With this approach, no attack processes can be injected in the traffic flow, and the samples are useful for anomaly detection and thus do not enable forecasting or diagnostics on adversarial activities.

The following proposals are all based on real attacks and in that respect are close to our approach. In [15], the authors present a cyber-physical testbed to test intrusion detection of critical manufacturing systems. In this case, the adversarial activity was limited to five specific attack vectors. Older proposals, such as [16,17], consider a limited range of attacks, and are not meant to be flexible tools as ours is.

The LARIAT [18] and TIDeS [19] emulators and the LLSIM [20] simulator are very interesting proposals. To the best of our knowledge, the projects have not been carried out.

To complete the picture, it makes sense to look at tools that emulate attacks, independently of their use in a testbed. Since we considered only open-source tools and chose to base our software on Metasploit, let us briefly consider rationales and alternatives.

A very good analysis of open-source tools that allow us to emulate attacks is in [21]: it explains very clearly the kind of analysis that drove our choice. The analysis in [21] takes into consideration, for each tool, parameters that are fundamental to us, like the coverage of MITRE techniques, ease of automation, and invasiveness. It points to Metasploit and MITRE's Caldera as the best tools for testing security software. Indeed, the requirements we laid out for the software (see Section 6) are met by both these software. We preferred Metasploit over Caldera since the latter requires installing agents on target machines, which

we deemed too invasive. We used Caldera to attack machines with a Linux OS, even though in [21] it is reported as a tool for Windows systems only.

Another noteworthy framework is AutoTTP [22], a scripting tool for automating complex attack procedures based on open-source attack emulators' APIs. It can automate Metasploit processes as well as Empire's (Empire [23] is a post-exploitation framework that is based on PowerShell for Windows and Python 3 for Linux and OsX). Since we want to customize the attack process based on the information on the network, devices, and applications (see Section 7), we cannot benefit from AutoTTP that requires specific scripting for each attack process.

Finally, we must mention Wattson [24], a co-simulation framework designed for implementing cyberattacks targeting power grids. Wattson enables the reproduction of both the fingerprint of such attacks on ICT communications and their impact on physical devices. It is built on Containernet [25] for network communication emulation and pandapower [26] for power grid simulation, coordinated by a synchronization layer to allow realistic representation of monitoring and control networks. Unlike our platform, Wattson focuses on reproducing IEC 60870-5-104 [27] communications. Although it supports executing individual attacks using built-in components or by invoking Metasploit scripts, it lacks a concept of attack paths for fully automated, multi-step attack execution.

3. Background

3.1. MITRE ATT&CK

The MITRE ATT&CK framework [28] is the backdrop of our work. It is a repository of adversarial activity, in which attack processes have been split into basic steps (*techniques*) that aim at achieving short-term goals (*tactics*). The tactics and techniques catalogue is a common reference for cybersecurity practitioners and allows for the adoption of a shared language for analysis, categorization, and mitigation of attacks. We mainly draw the information we need from the Enterprise Matrix [29], which focuses on attack methodologies related to desktop and cloud environments, and the ICS matrix [30], a specialized attack knowledge base for industrial control systems, which is very useful in the domain of critical infrastructures.

3.2. SecuriDN

SecuriDN [5] is a tool we implemented over the framework DrawNET Modeling system (DrawNET) [31]. It allows us to graphically define the architecture of an IT/OT system, including the attacks to which the system is possibly vulnerable and the parameters that characterize the various attack steps. This information is derived from the MITRE ATT&CK project. Its name recalls that it is a specialization of DrawNET.

We give in this section a concise introduction to both DrawNET and SecuriDN. A more detailed description can be found, e.g., in [5].

3.2.1. DrawNET

DrawNET allows us to design and solve models in any graph-based formalism. It is written in Java and is based on the DNlib library. It offers an editor for the design of new formalisms and models that will be saved in XML files. It has been used in various contexts (e.g., [32–34]).

The architecture of DrawNET contemplates three distinct levels. The starting point is a formalism. On a formalism a specific model can be built. Solvers can be defined to work on a model. We summarize this in Figure 1. All data, including graphical elements, are saved to XML files.

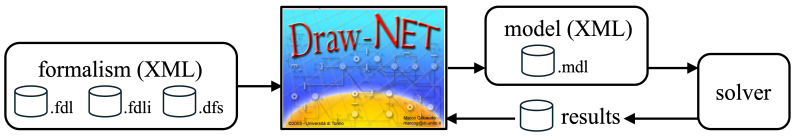

Figure 1. DrawNET general architecture.

Formalism Level: The formalism level specifies the primitives that will be required to design any model.

The fundamental entities are the elements (nodes and edges), since DrawNET works with graph-based models.

- Elements have attributes, which are called properties that are typed (integer, float, boolean, string, etc.); among them we have graphical properties, such as shape, size, and color.
- Elements must satisfy consistency relations, if any, expressed by constraints.
- Elements can be organized hierarchically, with elements containing sub-elements.

Model Level: DrawNET enables graphical design of models. To specify a model, it is first necessary to specify which formalism the model will instantiate. DrawNET loads the requested formalism and then provides the elements, as graphical objects, to instantiate, allowing us to set the properties and enforce the constraints and the element hierarchy that has been established in the formalism.

3.2.2. SecuriDN's Formalism and Model

Recall that it is possible to define a hierarchy of elements in DrawNET. One of the top-level elements of SecuriDN is the architecture graph. The others are the global attack graph (see Section 5.1) and the Dynamic Bayesian Network (DBN), which we will not discuss in this paper (see [5]).

Architecture Graph: The nodes of the architecture graph are assets and the edges are associations between assets. Assets represent all possible targets of attacks and, at present, include computers, networks, applications, channels, etc. The associations are used to relate the assets in that they specify which network a device is connected to, on which device an application is running, which applications communicate on a certain channel, and so on.

The constraints are imposed to keep the model consistent; for example, an instance of an application can run on a computer.

In the architecture graph, the operator will also specify an asset that has already been compromised by the attacker (which will be the origin of the attack process) to design scenarios based on hypotheses on more exposed assets. Moreover, the cybersecurity analyst will specify a target of the attack to study the security posture of specific assets.

An architecture graph is shown on the left side of Figure 2 (see Table 1 for an explanation of the symbols).

Table 1. Meaning of symbols in SecuriDN's architecture graph (see an example in Figure 2).

Icon	Meaning	Icon	Meaning
	network		firewall
	host		application
	IED		virtual environment
	compromised asset		communication channel
	analysis target		

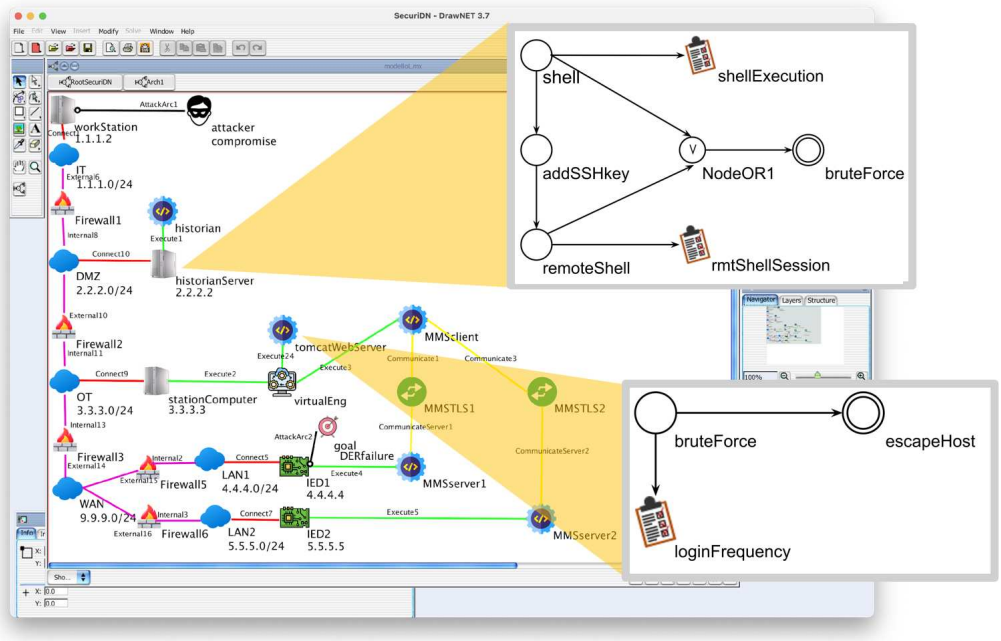

Figure 2. An architecture graph with local attack graphs of two assets. Link colors represent specific association types between assets (for further details refer to [5]).

Each host is connected to the network it belongs to. All applications are explicitly drawn and connected to the host on which they are executed. In this example one can also see communications channels (green icons). Notice the compromised workstation at the top of the figure and the target icon denoting the asset whose security posture is of interest in the current analysis. See Section 7 for an illustration of the example architecture.

Local Attack Graph. To each asset a local attack graph (locAG) is associated. The locAG expresses the attack processes that an attacker could potentially try against the asset. The nodes of the locAG implement techniques from the MITRE ATT&CK project and defense measures.

Figure 2 shows two simplified local attack graphs: the topmost for a historian server, and the other one for a Tomcat service running in the virtual engine on a station computer. See Section 7 for a description of the example architecture.

In both locAGs of Figure 2 no defenses appear. The techniques are represented by circles. The double circles have a special role; we will return to them in Section 5.1. The directed edges in the graph express preconditions when they connect techniques. A technique with outgoing edges enables its neighboring techniques. For instance, in the example of Figure 2, *remoteShell* can take place only after *addSSHkey* has been successfully exploited.

Techniques can be parameterized with additional information, depending on the specific way in which SecuriDN is applied. We are interested in the annotation of the target port for external techniques. Another use, out of the scope of this paper, is, for instance, the parameterization of the detection models.

When the edge's origin is a defense, it will connect the defense to a technique, representing a measure to mitigate that attack step (defenses are not taken into account in this paper).

Additional nodes are logical operators (AND, OR) that enable us to express more than one precondition to a technique or alternative preconditions, respectively.

In a locAG also *analytics* appear, i.e., units of a monitoring system that collect information useful to expose adversarial activities (these are inspired by the MITRE project CAR [35]). They are depicted as notepads in Figure 2. In this work, we will not discuss ana-

lytics, since the paper focuses on the attack processes (an example of analytics is reported in [5]).

A default locAG is provided for each asset, but new locAGs can be designed or imported from a library. Currently, locAGs must be built by an expert, but we are envisaging automatic generation as future work.

3.2.3. Solver Level

Solvers enable any processing of the model: conversions, analyses, simulations, etc. The solvers are written in Java, and the results of the processing can be displayed by DrawNET.

Among the solvers we implemented for SecuriDN, we will discuss in later sections one solver that derives the global security posture of the whole network from the locAGs (Section 5.1) and one used to produce attack processes for Netsploit (Section 5.2).

In addition, another solver produces a Dynamic Bayesian Network to be used as a detection model and an Octave [36] script to run it. We will not present this solver here, but it is fundamental to complete the context (again, we refer the interested reader to [5]).

3.3. Metasploit

Metasploit [37] is one of the best-known pen-testing tools. It is highly modular and offers support for reconnaissance, exploit, and post-exploit activity. According to [21], Metasploit was covering 100 MITRE techniques out of 266 existing when the paper was published, and 11 tactics out of 12 (although some had much better coverage than others). Metasploit does not provide pre-configured multi-step attack processes but accepts the definition of procedures through "resource scripts" written in Ruby, and a Python library allows programmatic access to Metasploit APIs.

4. The Complete Picture

The operational pipeline that translates network models defined by the cybersecurity analysts into emulated attack scenarios is a crucial part of a testbed for the design, deployment, and validation of detection models. The pipeline is illustrated in Figure 3 along with the additional functionalities of SecuriDN. The result is a realistic but controlled environment that mimics crucial aspects of power control infrastructure to evaluate the effectiveness of detection models.

Figure 3 shows step-by-step how SecuriDN is used and how it interacts with the validation testbed. In brief, the numbered arrows in the picture describe SecuriDN's functionalities that, from the definition of the architecture, allow us to derive attack processes and a detection model with its manager. The thick arrows labeled with letters show how SecuriDN's artifacts are integrated in the validation testbed.

The process starts when cybersecurity analysts use the graphical interface of SecuriDN to construct a detailed model of the target network (arrow (1) in Figure 3). This model includes the devices, the applications running on them, and the network topology; moreover, it also represents the network firewalls and their rules. In this way the analysts can incorporate hypotheses about potential weak points and identify critical assets on which they want to focus their analysis. In our example, we consider a possible ICT infrastructure integrating Distributed Energy Resources (DERs) where the resources are controlled by a utility infrastructure.

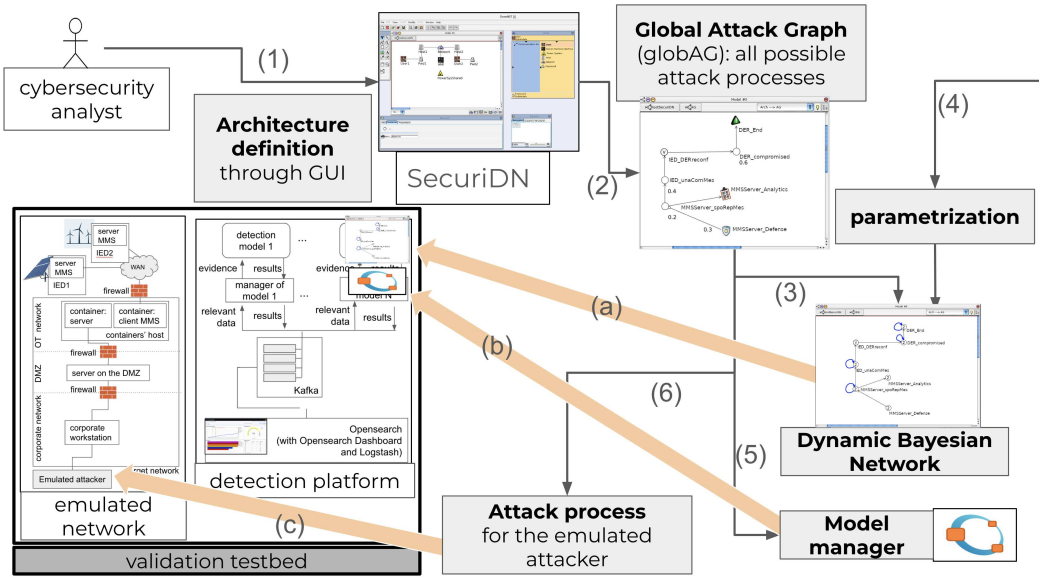

Figure 3. SecuriDN: functions (number labeled arrows) and integration in the platform (letter labeled arrows). The validation testbed is expanded in Figure 4.

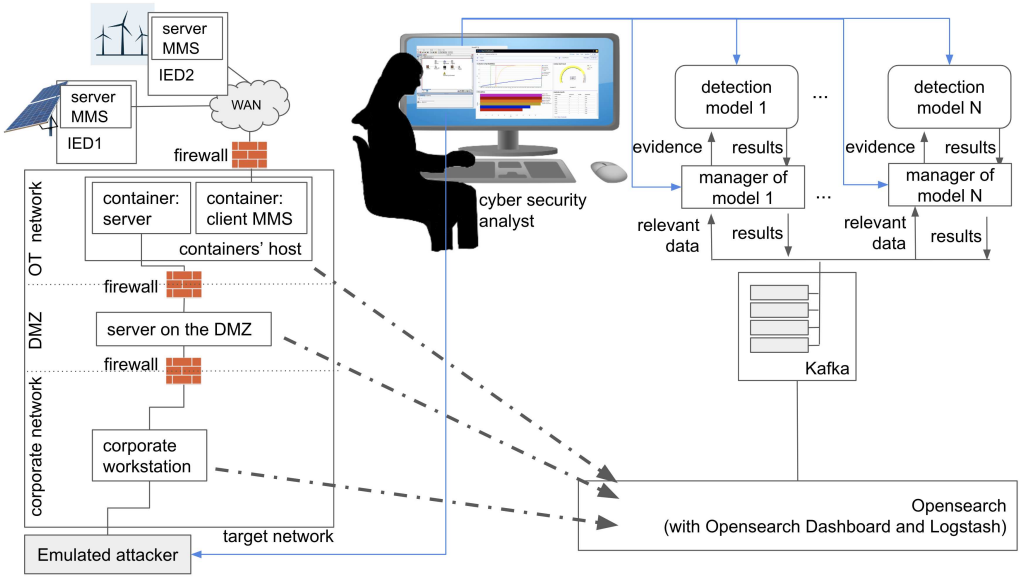

Figure 4. The logical architecture of the framework.

Based on the analysts' input, SecuriDN first associates each defined asset with the corresponding local attack graph (locAG). As we detailed in Section 3.2, these locAGs represent known attack techniques with their execution dependencies, derived from the MITRE ATT&CK framework. For instance, operational conditions affecting lateral movement and persistence attacks can be introduced by the analyst whenever needed. SecuriDN then composes individual locAGs based on the network topology, firewall policies, and software running on each device, as specified in the architectural definition in SecuriDN. This way it builds a global attack graph (globAG) that provides a general view of potential attack paths across the entire modeled network, from the specified possible attacker entry points to the predefined targets (arrow (2) in Figure 3).

In turn, the globAG serves as a base to create emulated attack sequences. A specific solver within SecuriDN (further discussed in Section 5.2) traverses this graph and produces an ordered list of attack techniques that represents a specific plausible attack campaign performed on the vulnerable network. The generation process is represented by arrow (6) in Figure 3.

The attack sequence is then provided to Netsploit, our automated adversary (as depicted by the thick arrow labeled (c) in Figure 3), to be executed within the emulated network. Figure 4 shows in detail the validation testbed from the lower left corner of Figure 3. The platform itself consists of containerized hosts orchestrated as a stack [38] via DockerSwarm [39].

Notice that since the target network and the attacks are emulated and not simulated, there could be unforeseen effects of the attack processes that cause unexpected behavior. This is a price to pay for a more realistic execution environment.

In any case, it is always the user's responsibility to ensure that SecuriDN configuration correctly represents the target system and to import/introduce the correct locAGs, and this is true also for the testbed and therefore the emulated network. If the configuration is correct, then the implementation of SecuriDN ensures coherence and compatibility.

Concurrently, a monitoring system based on Opensearch and Logstash (bottom-right in Figure 4) generates logs about specific monitored events happening within the emulated environment. This data is then fed, by an event streaming platform like Kafka, to a series of deployed detection models (top-right side of Figure 4). The alerts and diagnostic or predictive information that are generated by these models can then be compared against the ground truth of the emulated attack, allowing analysts to assess the detection mechanisms' performance. The monitoring system collects artifacts generated by real attacks, and it does so using standard monitoring tools such as `auditd` and the Elastic `beats`. Thus, it does not take any advantage from the fact that it is running on an emulated system or that the attacker is emulated and controlled by the security analyst. False positives and negatives can occur in the detection, but validating the detection models is precisely the purpose of the platform. While data collection and distribution and the detection models' capabilities have been detailed in [5], in this work we focus on the attack emulation pipeline that provides meaningful input for the validation of detection models. In [40,41] we provide more detail and background on the detection models, as well as an analysis of their effectiveness for monitoring in real time the security posture of the network and for planning adequate defenses and a useful monitoring system.

The right side of Figure 3 shows how SecuriDN also produces detection models and the software to run them (arrows (3) and (5) in the figure), that are deployed in the detection platform (thick arrows (a) and (b)), also discussed in [5], whereas their parameterization (4) uses additional techniques that are out of the scope of this paper.

5. From the Network to an Attack Process

To produce potential attack processes for Netsploit to emulate, we first need to create a global picture from the fragmentary security information provided by the locAGs. We show in this section how the global cybersecurity posture is determined and how an attack sequence is derived from it.

5.1. The Security Posture of the Whole Network

A solver of SecuriDN enables the integration of all local security information given by the locAGs to obtain global information about the network. We described an earlier version of this solver in [5], but since then the architecture graph has been enhanced with firewalls, and the solver has been modified accordingly. In particular, the addition of firewall modelling capabilities to the formalism enables a more flexible propagation of adversarial activity, at the same time bounding the computational cost of the generation process (see [42]).

The firewalls we have added are packet filters that implement a deny-all policy in which all traffic is rejected unless it is explicitly allowed. The whitelists on each firewall

specify the direction of the traffic (looking at the incoming and outgoing interfaces), the source and destination IPs, and the destination port. We assume that the packet filters are dynamic, i.e., carry out stateful inspection of the packets, and so we only need to specify acceptance of the packets that initiate a connection. Figure 5 shows an example in which we allow traffic from IP 2.2.2.2 (the historian server) to IP 3.3.3.3 (the station computer) if directed to port 443.

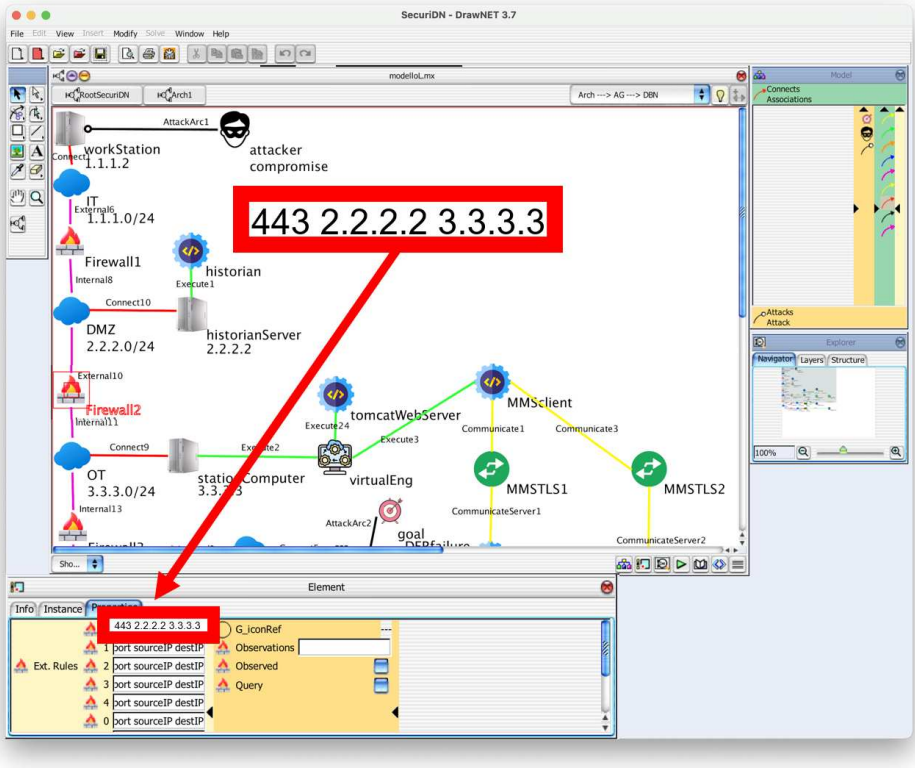

Figure 5. Example of firewall configuration from SecuriDN's interface.

To combine locAGs into a globAG, the special techniques depicted with the double circles in Figure 2 are key. We refer to those as *external techniques* as opposed to those with a single circle that are internal. The idea is that external techniques are subsequent attack steps that can be taken on some different assets, if reachable. Of course, the locAGs of the figure are highly simplified, since we will always assume that the attacker has multiple choices. In the figure, the external techniques are *bruteForce* and *escapeHost*. The number below *bruteForce* in Figure 6 is the number of the port targeted by the attack. For the technique *escapeHost*, no port number applies.

To fuse two locAGs into a single one, we need to match an external technique of the first to an internal one of the second. In the example shown in Figure 2, if the station computer vulnerable to a brute force attack can be reached from the historian that has a *bruteForce* external technique, the two locAGs can be combined into a single one.

SecuriDN looks for possible matches for external techniques by visiting the architecture graph, but in doing so, it respects firewall rules. So for instance, in our example, let us consider the search for an asset on which a *bruteForce* attack can be carried out. The parameter of the *bruteForce* node specifies a target port 443. As the search reaches the firewall that separates the DMZ from the corporate network (at the top of Figure 6), it aborts the search since the whitelist of that firewall does not allow any through traffic in this direction.

On the other hand, the whitelist of Firewall 2 in Figure 7 has a rule admitting traffic from the historian to the station computer directed to port 443.

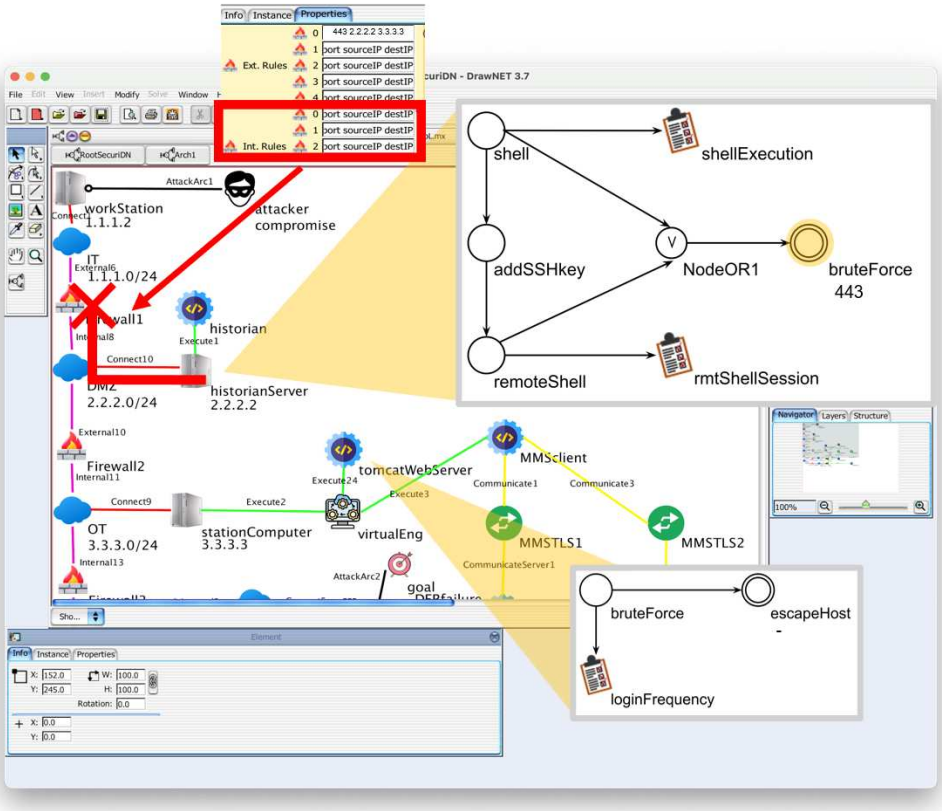

Figure 6. The firewall rules in the fusion of locAGs: no internal rule is matched in Firewall 1.

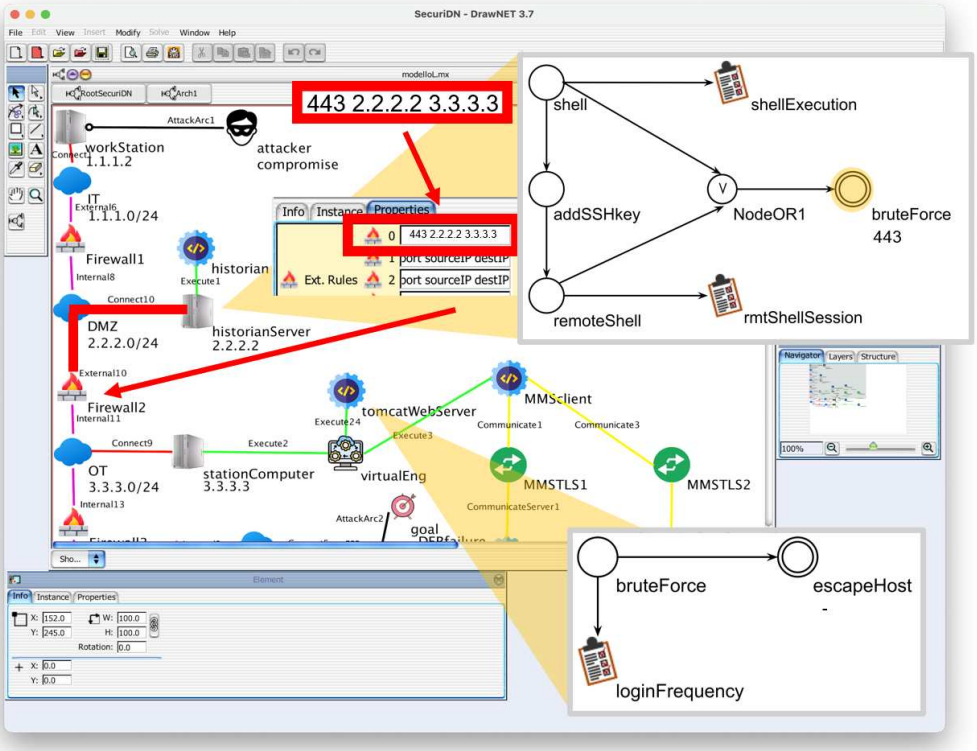

Figure 7. The firewall rules in the fusion of locAGs: an external rule inside Firewall 2 is matched.

So the search continues, but SecuriDN can only look for a *bruteForce* internal node in the locAG of the station computer. Finding the node, it fuses the two locAGs, matching their *bruteForce* external and internal nodes (Figure 8).

This process is iterated for all external nodes in each locAG. All possible locAGs are fused. The process might obviously produce multiple graphs, but recall that we focus on particular weakness hypotheses and on the security posture of specific assets. These focal points are defined through the SecuriDN interface as an asset already compromised by the attacker and a target of the adversarial activity, as we anticipated in Section 4 and detailed in Section 3.2.2.

Thus the globAG will be a directed acyclic graph that has the compromised device as a source and the target as a sink, pruning all dead branches that would not contribute to an attack process from source to sink.

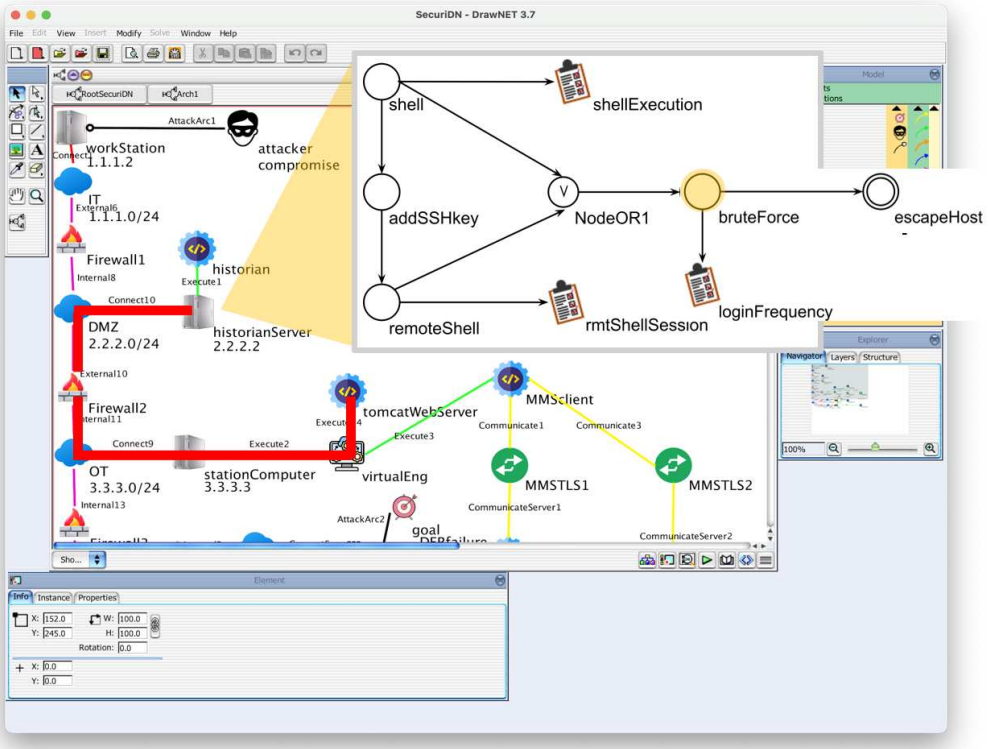

Figure 8. Fusion of two locAGs: the search finds an internal node (*bruteForce* in the Tomcat web server) matching the source external node (*bruteForce* within the historian server).

The globAG that we obtain is also used by the solver to derive the detection models (see [5]). But in the following, we focus on the generation of a schedule for adversarial activity.

Further details and an analysis of the algorithm can be found in Appendix A.

5.2. Attack Process Generation

Given the globAG, SecuriDN can generate an attack sequence that the attacker, Netsploit, executes in the emulated environment. A specific solver is in charge of this step. It saves the attack sequence in a JSON file.

The solver implements a variant (Algorithm 1) of Kahn's algorithm for the topological order [43] that processes the global graph respecting the attack preconditions and specifically enabling an OR node when a precondition is satisfied; the algorithm can be parameterized to use different strategies.

Algorithm 1 Generation of an attack sequence from the globAG

```
ready = ∅
attack_sequence = []                          ▷ attack_sequence is an ordered sequence
for v ← 1 to n do
    indegree[v] ← in-degree of node v
    if v is an OR node then
        isOR[v] = T
    end if
    if indegree[v] = 0 then
        ready ← ready ⋃ {v}
    end if
end for
while ready is not empty do
    v ← a node from ready                     ▷ v is removed from ready
    attack_sequence ← v                       ▷ v is added to the end of attack_sequence
    for v′ neighbour of v such that indegree[v] > 0 do
        if isOR[v′] then
            indegree[v′] ← 0
        else
            indegree[v′] − −
        end if
        if indegree[v′] = 0 then
            ready ← ready ⋃ {v′}
        end if
    end for
end while
```

The choice of the order of attack steps in the algorithm can be randomized, still respecting the preconditions. The resulting process can be saved and reused; this is an interesting feature that enables reproducibility of each attack campaign.

Correctness and complexity of the algorithm are analyzed in Appendix B.

If the network is correctly modeled in SecuriDN and MITRE techniques are correctly modeled and implemented, the attack sequences output by the solver can be emulated.

6. The Attacker

The attack sequence produced by SecuriDN is then run by our automatic adversary, Netsploit.

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 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 increasingly specializing our tool for power systems.

We opted for building Netsploit over well-established open-source software to provide a tool that can be of use to the community at large.

Our criteria for choosing the core software on which to base our automated attacker include the necessity of a tool that achieves the following:

1. Implements many MITRE techniques;
2. Is easy to automate, i.e., it offers a command line interface or APIs;
3. Allows Linux systems as targets;
4. Enables real, life-like attacks;
5. Is as little invasive as possible, meaning that it does not require installing software on the victim machines;
6. Is open source.

We built our adversary around Metasploit, which has become the toolbox of our adversary. The choice has been discussed in Section 2, taking advantage of the very thorough survey [21].

We also added for Netsploit the capability of running attack processes that do not use the Metasploit module. This has become necessary since we want to specifically focus on the power grid and must therefore implement more specific attacks. For instance, we implemented a Man-in-The-Middle attack against the protocol Manufacturing Message Specification (MMS) over TLS, the communication protocol used by the DERs to communicate with the controller. This attack is not among Metasploit's modules, and so it was completely implemented in our project in Python.

We have also made Netsploit able to carry out random attacks, since this is interesting from the defense point of view, allowing adversarial activity not entirely under control. But the reality is that we are rather focusing on having our attacker follow a specified attack sequence so we can have full control of its activity to test the response of the detection models.

6.1. Attack Database

The attacker software executes attacks from a database. The attack database is a JSON file that lists attacks, identified by a unique name, with the necessary information to run them.

The information on each attack step includes instructions, configuration information, and additional useful information on the attack.

The instructions are mostly lists of Metasploit directives that generally also include a call to a resource script written in Ruby, allowing us to specify more complex attacks. Some attacks also use software external to Metasploit.

Each attack requires configuration files, possibly with different structure, especially for more complex attacks that need to download a post-exploit payload. The record for each attack specifies the path to the templates for the configuration files and where the completed configuration files must be stored. On these files, the fields that specify the attack target and some custom ports are set up at run time depending on the chosen victim. We will discuss in Section 6.3 how this information is retrieved.

A wait time is also specified for each attack. This is customized when designing the attack, generally to leave Metasploit enough time to establish a new session with the victim or for other concurrent actions to take place. This time-out depends on the specific attack.

The attack type specifies whether the attack uses a single Metasploit module, or a resource script (with possibly some preliminary instructions), or an attack carried out with specific software that does not call Metasploit. A final type of attack, *NotImplementedAttack*, is useful for testing purposes; when coupled with the attack sequence provided by SecuriDN, it means that the attack is a dummy. In this case, all other fields are empty.

6.2. Managing Sessions

This is a tricky point since Netsploit must interact with Metasploit. We especially need Netsploit to be able to move across the network with lateral movements to progress towards its final target. So successful exploits open Metasploit sessions on the victim machine, and the session is upgraded to a Meterpreter session (see [44] for an introduction to Meterpreter and [45] for up-to-date information) to obtain the full power and stealth of Metasploit. Exploits that must open a shell are considered successful by the program only if the new session has been opened.

In some situations, typically when we are traversing firewalls, to attack the next target, we might need to carry on the subsequent attack steps from the foothold we have most recently established. To do this, we pivot on the latest victim implementing port forwarding. This way we mimic the behavior of an attacker that would move in depth into the network, progressing from the latest device it has compromised.

6.3. Configuration

Netsploit needs all attacks to be configured for the right environment; that is, the correct IPs and ports must be specified. This way the framework can be used to emulate networks with their real IPs for more clarity and ease of use by the cybersecurity analysts.

Moreover, we do not want to constrain attacks to be carried out only against designated machines. We want to be able to change the topology of our testbed network, adding and moving devices, including firewalls, and changing rules on the firewalls. This implies that the attacker's strategy must be flexible as well.

To obtain this, our requirements are that each attack can be carried out against any machine (even if it fails) and against services that might not use standard ports.

The information for the attacks is obtained by Netsploit from two separate configuration files. One is the JSON attack sequence generated by SecuriDN; the other is the Docker stack configuration file. At present, consistency between these two files must be maintained manually, but we plan to add a consistency checker.

The stack configuration file specifies on which virtual machines the Docker services must run and other information useful for running the stack. But it also specifies service ports that are to be used by Netsploit (or more precisely, by Metasploit), since these ports are related to the emulated attacker's own strategy rather than to the planned attack sequence, in the sense that those ports are independent from the specific adversarial process.

The attack sequence file lists all attacks, identifying them by the names of the techniques they implement, as in the attack database. It also lists the ports on which the attack must be carried out. Notice that this information, which is produced by SecuriDN, is based on the information on the reference network that is input by the cybersecurity operator through SecuriDN's graphical interface.

Attacks are configured on the fly by Netsploit before execution. The configuration file templates are retrieved at the location specified in the database of attacks, and all placeholders are filled in with information from the attack sequences generated by SecuriDN and from additional information in the Docker stack configuration file.

7. An Example

In this use case of SecuriDN, we defined the network model of Figure 9, which has the same architecture as the emulated network on the left-hand side of Figure 4. This network is part of the ICT infrastructure for the monitoring and control of power grids integrating Distributed Energy Resources. In this example, we model only some of the assets for brevity, but the system is useful in a real-life application only if all assets are modeled, along with their locAGs (see Section 8 for a discussion on complexity issues). See again Table 1 for an explanation of the symbols in Figure 9. Here the corporate network of a power distribution utility is at the top, with network address 1.1.1.0/24, the DMZ is at the center, with network address 2.2.2.0/24. Below the DMZ, the OT network is depicted, with network address 3.3.3.0/24. At the bottom, the WAN connects the OT network to the Distributed Energy Resources (DERs) with IEDs 1 and 2.

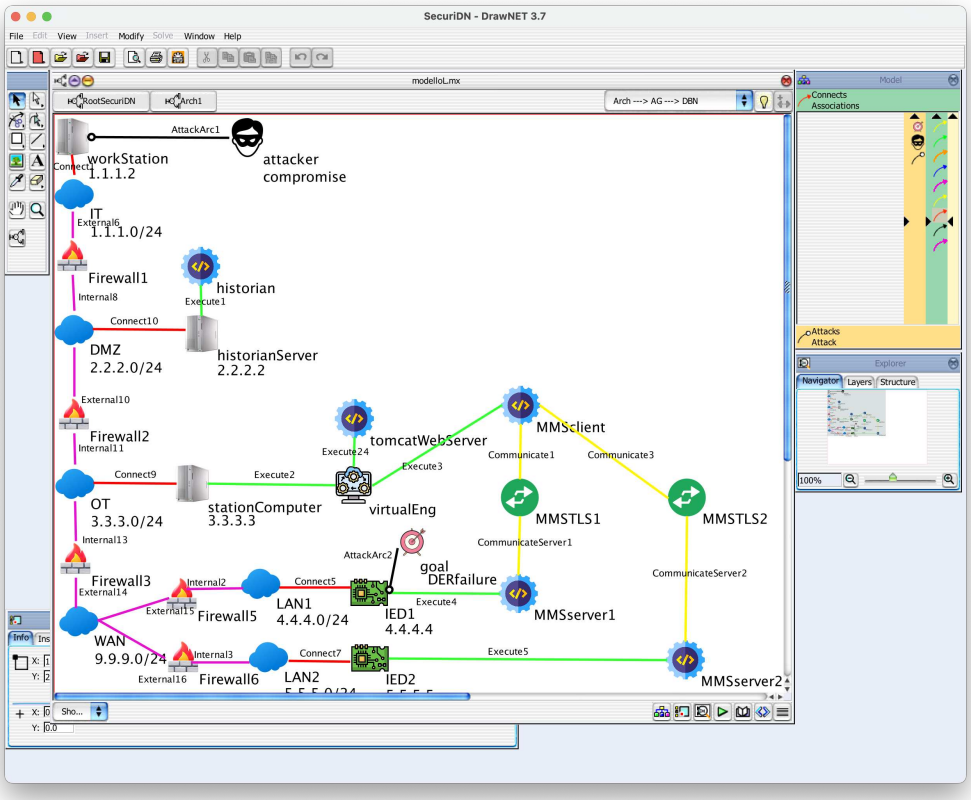

Figure 9. Example network as defined in the architecture graph of SecuriDN.

Notice how the relevant applications are defined: the historian on the historian server; the virtual environment on which a Tomcat server runs on the station computer; and on the same host, the MMS client; in addition, the MMS servers on the IEDs.

The historian database is updated from the OT network, but it can be consulted from the IT network through a web interface that runs on the same host. The web interface allows us to consult historian data from the corporate network, so Firewall 1 allows HTTPS traffic from network 1.1.1.0/24 to the historian server on port 443, but no traffic initiated from the DMZ. The access to the historian is read-only, so no authentication mechanism is implemented. The lack of authentication, although motivated by the read-only configuration, opens up a scenario in which attackers, being allowed to interact with the web interface, remotely exploit vulnerabilities of the system (see later). The historian data is updated from the OT, and Firewall 2 allows this traffic.

Firewall 2 also allows HTTPS traffic to any server in the OT. This configuration was planned because of a second host on the DMZ, the certificate management server, that runs a PKI application and connects to the target hosts through HTTPS connections. We have not explicitly included this host in the network in SecuriDN, to keep it simpler, but have kept the necessary rules in the firewall's whitelist.

The MMS client on the station computer connects to DERs to request measures and to issue commands. Corresponding rules on the whitelists of firewalls 3, 4, and 5 allow this traffic.

The analysts have assumed the workstation on the corporate network as a potential weak point, compromised by the attacker: this is shown by the attacker icon at the top of Figure 9. They want to analyze the security posture of IED1, as specified by the target icon in the figure.

Each asset has a locAG as explained in Section 3.2.

We base the choice of techniques on the MITRE ATT&CK project, and in general the idea is to propose a coverage that is as complete as possible (see Section 8). In the example we propose and in the testing of the platform, we started by choosing techniques and implementations that have the capability to disturb the data flows of energy control systems, potentially producing a higher impact on the physical system.

The locAG of the historian application is shown in Figure 10a. The attack steps are as follows:

- *scanVuln* that implements the technique *Active Scanning* for tactic Reconnaissance from the Enterprise ATT&CK Matrix.
- *reverseShellInjection* that implements the technique *Exploitation of Remote Services* for tactic Lateral Movement from the ICS ATT&CK matrix.
- *shell* is an external technique. Recall the function of external techniques explained in Section 5.1: it will be merged with a technique of the same name on the locAG of a different asset, if reachable.

Figure 10b shows the locAGs of the historian server. The techniques considered here are as follows:

- *shell* implements the technique *Command Line Interface* for the tactic Execution of the ICS ATT&CK matrix. The precondition for this attack is that the attackers have user access on the target host. They obtain a shell on the host. When building the globAG, this technique will be merged with the external *shell* from the locAG of Figure 10a.
- *addSSHkey* implements the technique *Account manipulation* for the tactic Persistence of the Enterprise ATT&CK Matrix: An attacker, having obtained a shell as user U on a host that runs SSH, will install an SSH public key for persistence.
- *remoteShell* implements the technique *Remote Access Software* for the tactic Command and Control of the Enterprise ATT&CK Matrix: A remote SSH shell is opened on the victim.
- The external technique *bruteForce*.

In the locAG of the Tomcat server in Figure 10c, only one attack step is considered. This is *bruteForce*, an implementation of *Unsecured Credentials* for tactic Credential Access of the Enterprise ATT&CK Matrix: it is a brute force attack to guess a password. In addition, there is the external technique *escapeHost*.

Figure 10d shows the locAG of the virtualization engine that runs on the station computer. Here again we consider a single possible attack step, *escapeHost*, that implements the technique *Escape to Host* for tactic Privilege Escalation from the Container ATT&CK matrix, which enables the attacker that has taken control of a container to overcome the container limitation and take control of the host the container is running on. The external technique is *shell*.

For the sake of simplicity we limit our description to these four locAGs, because this is sufficient to exemplify our work.

Figure 11 shows a fragment of the globAG in which it can be seen how the four locAGs we discussed are fused.

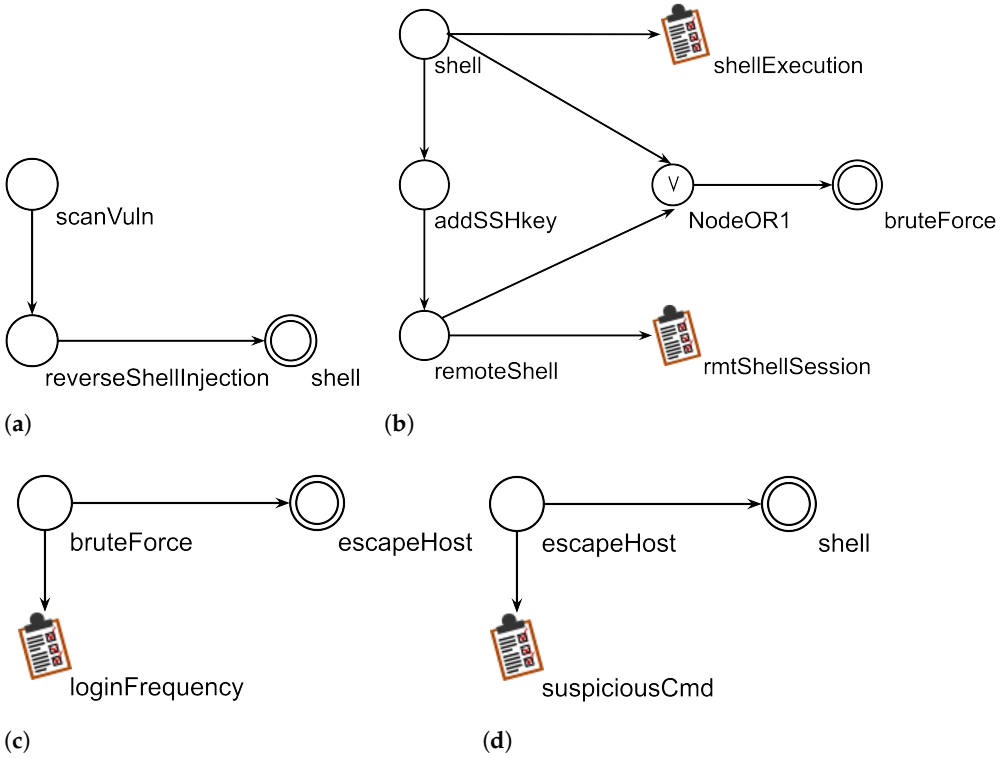

Figure 10. Local attack graphs of the historian application (**a**), the historian server (**b**), the Tomcat server (**c**) and the virtualization engine (**d**).

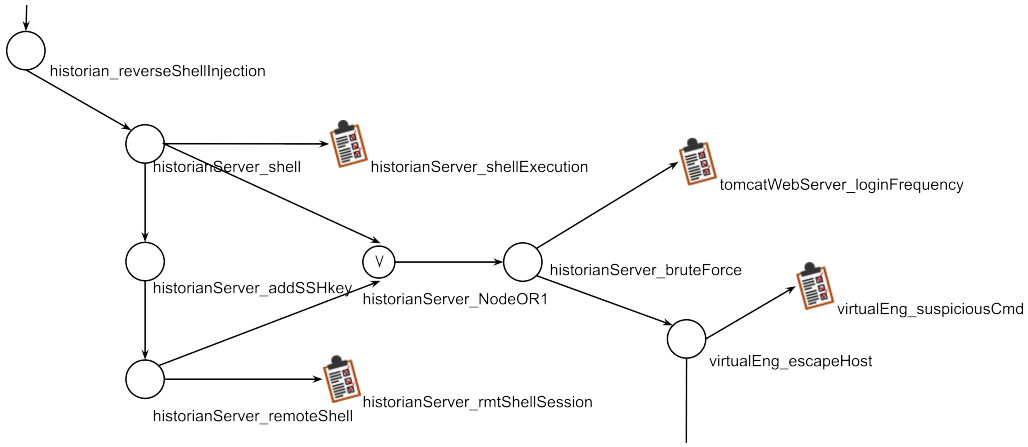

Figure 11. Fragment of the globAG.

From this globAG, SecuriDN derives an attack sequence. Listing 1 shows the fragment of attack sequence derived from the locAGs we have discussed above.

The first item in the sequence, with attribute "compromise", indicates the attackers' initial foothold, whereas the last one, with attribute "target", denotes the asset targeted by the current attack process and focus of the analysis. These two entries are irrelevant for the attack and are included for documentation. We omitted other attack steps that are generated from locAGs we have not discussed in this work.

The attack steps in the attack sequence are taken from the attack database, which specifies the implementations. For instance, *reverseShellInjection* is an SQL injection on the historian web interface. The attack opens a reverse shell from the historian server to the workstation compromised by the attackers. It is successful if the web application does not sanitize inputs and the User Defined Function `sys_exec()` is available on the SQL database.

In our environment, moreover, we assume that MySQL runs as root. The database entry for this attack step is in Listing 2.

Listing 1. Fragment of an attack sequence, where "..." replaces the omitted part.

```
{
  "attack_sequence":[
    {
     "attack_name":"compromise",
     "IP":"1.1.1.2",
     "additional_attribute":"Attacker"
    },
    {
     "attack_name":"scanVuln",
     "IP":"2.2.2.2",
     "additional_attribute":"None"
    },
    {
     "attack_name":"reverseShellInjection",
     "IP":"2.2.2.2",
     "PORT":"443",
     "additional_attribute":"None"
    },
    {
     "attack_name":"shell",
     "IP":"2.2.2.2",
     "additional_attribute":"None"
    },
    {
     "attack_name":"addSSHkey",
     "IP":"2.2.2.2",
     "PORT":"443",
     "additional_attribute":"None"
    },
    {
     "attack_name":"bruteForce",
     "IP":"3.3.3.3",
     "PORT":"443",
     "additional_attribute":"None"
    },
    {
     "attack_name":"escapeHost",
     "IP":"3.3.3.3",
     "additional_attribute":"None"
    },
    ...
    {
     "attack_name":"DERfailure",
     "IP":"4.4.4.4",
     "additional_attribute":"Target"
    }
]}
```

The database entry specifies the code for the attack (field "instructions"), the kind of attack (field "attack_type"), and more information required to configure correctly the script for the current environment. Additional information includes a "wait_time" (see Section 6.1), locations of configuration files, locations of templates and destinations for all necessary files, ports and IP addresses, and additional parameters for Metasploit (in this case "workspace" and "targeturi").

Listing 2. Entry for *reverseShellInjection* in the attack database.

```
"reverseShellInjection": {
        "instructions": "back \n resource /data/attacker/
            custom_attacks/sql_injection/reverse_shell_injection.rc",
        "attack_type": "ResourceAttack",
        "wait_time": "80",
        "config": "../stack/data/attacker/custom_attacks/
            sql_injection/config_rc.json",
        "templates": [
          {
            "template": "templates/reverse_shell_injection/config_rc.
                json",
            "output_path": "../stack/data/attacker/custom_attacks/
                sql_injection/config_rc.json",
            "fixed_params": {
              "workspace": ".",
              "targeturi": "/employees_search-1.0-SNAPSHOT/search"
            },
            "ports": {
              "lport": "reverse_shell_lport",
              "netcatport": "reverse_shell_netcat"
            },
            "attacker_ip": "{{lhost}}"
          }
        ]
      }
```

8. Discussion

8.1. Whitelists

Firewall rules are modeled using unique whitelists. Regardless of the fact that any set of rules can be expressed using only whitelists, it is important that the security analyst can configure SecuriDN mirroring exactly the rules that are implemented on the real firewall. This way there is a guarantee that SecuriDN is precisely aligned to the modeled system. Therefore, in the future, we need to enable more flexible ways to define firewall rules. This will not introduce conceptual nor complexity issues, since SecuriDN parses the list of rules much like a firewall does. If blacklist rules are introduced and the solver finds a drop rule that matches the parameters of the connection that is being considered, the search is directly interrupted.

8.2. Completeness of Vulnerability Information

Ideally, all attack techniques deemed plausible should likely be modeled in the locAGs to achieve significant results, leaning on the MITRE ATT&CK database for complete coverage. However, there is always the risk of incomplete modeling due to human error or undiscovered vulnerabilities. Moreover, complete coverage might not be feasible because of the excessive complexity it would entail. It is true that any missing (or wrong) information in the configuration of the tool will result in a less than optimal performance; therefore, continuous revision and update of the models are necessary. Finally, the tradeoff between complexity of the system and attack coverage should be taken carefully into account, the risk level that is deemed acceptable should be assessed, and techniques should be prioritized accordingly, based, e.g., on their frequency or impact.

This kind of approach is necessary with any cybersecurity tool, whether for assessment, detection, or protection, and they must always be viewed as a component of a more comprehensive cybersecurity plan. In particular, our framework, as any assessment tool, must be viewed as a support to security analysts and not as a definitive solution.

8.3. Complexity

Our algorithms are efficient, and we are using heuristics to make them even more responsive in practice; moreover, we are presenting here a prototype, and engineering of the software would definitely improve its performance. But it is a fact that a huge system with thousands of assets is not easy to deal with. The task is more difficult when the system is complex and heterogeneous, with a large choice of software running on its hosts and different policies and security levels required in different areas.

The solution to this is partitioning the system into smaller and more homogeneous units (again, modeling firewalls takes us in this direction [42]), observing each one in isolation but foreseeing a coordination of the single partitions with a hierarchical approach, a suitable exchange of information, and prompt inter-subsystem responses. The smart grid is indeed a System-of-Systems (SoS) [46], and there is a lot of research on modeling an SoS for cybersecurity assessment and protection (see, e.g., [47]).

9. Conclusions and Future Work

The resilience to cyber attacks of energy infrastructures is a challenging research area. Making the system resistant to an evolving threat landscape requires the deployment of advanced monitoring architectures able to collect security-related information through appropriate smart agents that feed powerful detection modules. The paper addresses the above challenge by means of an attack emulation tool—Netsploit—characterized by two innovative features: the tool implements attack processes automatically generated by SecuriDN by incorporating knowledge about attack techniques and defense measures, and it is integrated with an experimental monitoring and detection platform. The combination of these distinctive properties allows exploiting the tool for improving the detection capabilities of SecuriDN models and enhancing the monitoring and detection platform in real energy infrastructures.

Netsploit is being actively developed to make it daily more modular and flexible.

We plan to add logic to verify the consistency of the attack sequence with the Docker stack configuration file. This should not be blocking, because we might also want to have the emulated attacker fail or target nonexistent IPs, but we could detect unwanted misalignments of the emulated environment with the network defined on SecuriDN. Notice that this would be an extra for the simulated environment, in the sense that in general it is the responsibility of SecuriDN's users to make sure that the configuration matches the target network, as noted in Section 4.

At present, Netsploit determines the success of an attack when it sees that a new session has been established. Some attacks, like for instance those that pry into private data or install a private SSH key for persistence, do not establish new sessions. Therefore currently we have no way of verifying the correct execution of these attack steps from Netsploit. We plan to lean on the Metasploit internal database to communicate to Netsploit the attack success. Then Netsploit will access the record and determine if the attack has been successful.

To further extend this framework, we plan to apply our automated attack generation and execution approach on emulated networks of containers using tools such as Containernet-WiFI [48]. This framework enables the deployment of containers in resource-constrained environments, allowing us to evaluate the outcome of cyberattacks on legacy or low-power devices commonly used in decentralized monitoring and control infrastructures. Additionally, since the current version of our emulation platform only connects multiple host interfaces modeling ideal communication channels, the integration of Containernet-WiFi would allow us to create software-defined networks of multiple routers, switches, and access points with predetermined link properties such as data rate, delay, or packet

loss probability, which would greatly improve the emulation realism in terms of network-induced latencies.

Another possible extension would concern the evaluation of the bias introduced in the DBN model automatically generated by SecuriDN. Since both the DBN model structure and its parameters are derived by the globAG topology and the estimated technique completion time, respectively, it is crucial to verify that all attack dependencies are represented and that the resulting inferences adequately assess the attack progression in a real-world environment.

Moreover, we aim to broaden the attack methods currently available in Netsploit. For instance, deep learning-based soft sensors used in industrial control systems are vulnerable to adversarial transfer attacks [49]. This specific attack can be viewed as a concrete implementation of the abstract "Spoof Reporting Message" technique found in the MITRE ATT&CK database. It leverages deep neural networks to generate adversarial samples, causing the production of highly confident yet incorrect measurements. Such attacks would be a valuable addition to future extensions of Netsploit.

Author Contributions: Conceptualization, D.C., D.C.R., and L.E.; methodology, D.C., D.C.R., and L.E.; software, D.C., D.C.R., and L.E.; validation, D.S.; data curation, D.C., D.C.R., L.E., and D.S.; writing—original draft preparation, D.C., D.S., and L.E.; writing—review and editing, D.C., D.C.R., G.D., L.E., G.F., L.P., D.S., and R.T.; visualization, D.C.R. and L.E.; supervision, D.C. and L.E.; project administration, D.C. and R.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work is original and has been developed by a joint collaboration between RSE S.p.A. and the Università del Piemonte Orientale. This work has been partially financed by the Research Fund for the Italian Electrical System under the Three-Year Research Plan 2025–2027 (MASE, Decree No. 388 of 6 November 2024), in compliance with the Decree of 12 April 2024. We also acknowledge partial funding by "Bando Ricerca UPO 2022" in the framework of the project "Cybersecurity Risk Governance in Public Administration—CybeR-GoPA" (ID: 1083758, CUP: C15F21001720001). This work has the financial support of the Università del Piemonte Orientale.

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Acknowledgments: Netsploit and SecuriDN's solver that generates attack sequences are the result of activity from several students at UPO working under our supervision. In particular, we acknowledge the work of Luca Binotti, Simone Conti, Matteo Cosentino, and Gaia Sismondo. The authors would also like to thank Alberto Livio Beccaria for implementing DNlib library, and Marco Gribaudo for the development of DrawNET.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
AutoTTP	Automated Tactics Techniques and Procedures
ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
CAR	Cyber Analytics Repository
DBN	Dynamic Bayesian Network
DER	Distributed Energy Resource
DMZ	Demilitarized Zone
DNlib	DrawNET Library
globAG	Global Attack Graph
HTTPS	Hypertext Transfer Protocol Secure

ICS	Industrial Control System
IED	Intelligent Electronic Device
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation
LARIAT	Lincoln Adaptable Real-Time Information Assurance Testbed
LLSIM	Lincoln Laboratory Simulator
locAG	Local Attack Graph
MITRE	Massachusetts Institute of Technology Research and Engineering
MMS	Manufacturing Message Specification
OS	Operating System
OT	Operational Technology
SoS	System-of-Systems
SQL	Structured query language
SSH	Secure Shell
TIDeS	Testbed for Evaluating Intrusion Detection Systems
TLS	Transport Layer Security
WAN	Wide Area Network
XML	Extensible Markup Language

Appendix A. Analysis of the Merge Algorithm

Appendix A.1. Correctness

To prove the correctness of the algorithm, we must show that globAG contains all and only the paths from the compromised asset to the target that exploit the local attack processes described by the locAGs.

The algorithm first creates a single graph whose components are all the locAGs. Then, for each external technique, it visits the architecture graph looking for matching internal techniques. The search space is reduced by the rules on the firewalls: when the algorithm encounters a firewall, it continues past it only if the firewall rules allow it.

If it finds a single asset with the correct internal technique, it merges the external node with the internal one. If it finds multiple suitable assets, it clones the external technique and its incoming edge to have one external node for each asset and then merges each external/internal node pair.

On the other hand, if there are multiple external nodes matching the same internal technique, then an OR node is inserted before the internal technique, and the external nodes are both merged onto the OR node.

This procedure guarantees that each path from the compromised asset to target is in the graph obtained. Moreover, since firewall rules are respected, only paths that can be actually followed by the attacker are in the graph.

The final phase of the merge algorithm is the pruning.

Before the actual pruning, a depth-first visit of the graph, starting from the compromised asset, determines which nodes are relevant, i.e., belonging to a path from attacker to target.

If the visit reaches the target, then all ancestors of the target node in the visit are marked as relevant (the marking is performed as the control returns to the ancestors on the function call stack). During a depth-first search of a directed acyclic graph, while visiting a new node, one can find the following:

- A forward edge to a previously visited node: in this case the node currently visited is an ancestor of the head of the edge and thus if the head was marked as relevant, so was the tail;

- A transversal edge to a previously visited node: in this case, if the head of the edge is marked as relevant, also the current node and all of its ancestors must be marked relevant.

This ensures that all nodes belonging to a path from attacker to target are marked relevant during the visit, and so during the actual pruning, no attack processes are lost.

At this point, all nodes that are not marked relevant are removed from the output graph.

The pruning of the graph reduces the size of the globAG, limiting the costs of the solvers that process it further. For instance, the extraction of attack sequences from the graph will benefit from the pruning.

Appendix A.2. Costs

We analyze the complexity of the algorithm, assuming that the whitelists of the firewalls admit all through traffic. This gives us the asymptotic worst-case cost. The presence of more restrictive firewall rules reduces the search space and thus the costs, but it is a heuristic.

The first merge phase uses a hybrid visit, which transverses at the same time on the graph of all locAGs and the architecture graph. This amounts to visiting a graph with n_l nodes (the sum of the nodes of all the locAGs) and $m_l + m_a$ edges (the sum of the edges of all the locAGs and of the edges of the architecture graph). The visit has a linear cost in the dimension of the graph and must be repeated for each of the external techniques (whose number is bounded above by n_l), yielding a total cost of $O(n_l(n_l + m_l + m_a))$.

The pruning phase has a linear cost.

Appendix B. Analysis of the Solver That Produces Attack Sequences

The algorithm has the following properties:

1. It terminates.
2. All nodes are added to the ordered *attack_sequence*.
3. The topological order is respected, except for OR nodes. More precisely, the topological order is respected as follows:
   - For each edge (u, v) such that v is not an OR node, u comes before v in the *attack_sequence*;
   - If v is an OR node, it appears in *attack_sequence* after at least one node u such that (u, v) is an edge of the graph.
4. If the graph is represented using an adjacency list and choosing a node from the set *ready* has cost $O(1)$, the cost of the algorithm is $O(n + m)$, where n is the number of nodes in the graph and m the number of edges.

Notice that Properties 2 and 3 guarantee that the *attack_sequence* is as required. Indeed, all nodes will be in the sequence, and the topological order will be respected except in the case of OR nodes that can appear in the sequence as soon as one of their preconditions is met. Moreover, notice that picking nodes from the set *ready* at random allows us to randomize the attacker's strategy, respecting preconditions of attack steps.

Proof. We prove the following properties:

1. Each node enters *ready* exactly once, either in Line 9 if its indegree is zero, or when the value of *indegree* for that node becomes zero (Line 22). Since at each iteration of the while loop a node is removed from *ready*, the algorithm terminates after one iteration per node.
2. Since each node enters the set *ready* by Property 1 and it is appended to the *attack_sequence* in Line 14, this property also holds.

3. If node v is not an OR node, its *indegree* becomes zero only after all of the nodes u such that (u, v) is an edge of the graph have already been appended to the *attack_sequence* (Lines 14 and 19); therefore, the first part of this property holds. When v is an OR node, at least one of the nodes u of which v is a neighbor enters the *attack_sequence* before v, by Lines 14 and 17, and so also the second part of the property holds.
4. The initialization loop scans the whole graph, and therefore it has a cost of $O(n + m)$ if the graph is represented with an adjacency list. The main loop executes n times and all in all considers each edge once. The strategy for choosing a node from the set *ready* is not specified, and it could have a non-constant cost; but if its cost is constant, the bound holds.

This completes the proof. □

References

1. Langner, R. Stuxnet: Dissecting a Cyberwarfare Weapon. *IEEE Secur. Priv.* **2011**, *9*, 49–51. [CrossRef]
2. Khan, R.; Maynard, P.; Mclaughlin, K.; Laverty, D.; Sezer, S. Threat Analysis of BlackEnergy Malware for Synchrophasor based Real-time Control and Monitoring in Smart Grid. In Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR), Belfast, UK, 23–25 August 2016. [CrossRef]
3. Salazar, L.; Castro, S.R.; Lozano, J.; Koneru, K.; Zambon, E.; Huang, B.; Baldick, R.; Krotofil, M.; Rojas, A.; Cardenas, A.A. A Tale of Two Industroyers: It was the Season of Darkness. In Proceedings of the IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–23 May 2024; pp. 312–330. [CrossRef]
4. Stouffer, K.; Pease, M.; Tang, C.; Zimmerman, T.; Pillitteri, V.; Lightman, S.; Hahn, A.; Saravia, S.; Sherule, A.; Thompson, M. *Guide to Operational Technology (OT) Security*; Technical Report SP 800-82 Rev. 3; NIST: Gaithersburg, MD, USA, 2023. Available online: https://csrc.nist.gov/pubs/sp/800/82/r3/final (accessed on 29 September 2025).
5. Cerotti, D.; Codetta Raiteri, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Savarro, D.; Terruggia, R. SecuriDN: A Modeling Tool Supporting the Early Detection of Cyberattacks to Smart Energy Systems. *Energies* **2024**, *17*, 3882. [CrossRef]
6. Cerotti, D.; Codetta Raiteri, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Terruggia, R., A Modular Infrastructure for the Validation of Cyberattack Detection Systems. In *Power Systems Cybersecurity: Methods, Concepts, and Best Practices*; Haes Alhelou, H., Hatziargyriou, N., Dong, Z.Y., Eds.; Springer International Publishing: Cham, Switzerland, 2023; pp. 311–336. [CrossRef]
7. Ukwandu, E.; Farah, M.A.B.; Hindy, H.; Brosset, D.; Kavallieros, D.; Atkinson, R.; Tachtatzis, C.; Bures, M.; Andonovic, I.; Bellekens, X. A review of cyber-ranges and test-beds: Current and future trends. *Sensors* **2020**, *20*, 7148. [CrossRef] [PubMed]
8. Gunathilaka, P.; Mashima, D.; Chen, B. Softgrid: A software-based smart grid testbed for evaluating substation cybersecurity solutions. In Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy, Vienna, Austria, 28 October 2016; pp. 113–124.
9. Tsai, P.W.; Yang, C.S. Testbed@ TWISC: A network security experiment platform. *Int. J. Commun. Syst.* **2018**, *31*, e3446. [CrossRef]
10. Conti, M.; Donadel, D.; Turrin, F. A survey on industrial control system testbeds and datasets for security research. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 2248–2294. [CrossRef]
11. Blazek, P.; Fujdiak, R.; Mlynek, P.; Misurec, J. Development of cyber-physical security testbed based on IEC 61850 architecture. *Elektron. Elektrotechnika* **2019**, *25*, 82–87. [CrossRef]
12. Karch, M.; Rösch, D.; André, K.; Meshram, A.; Haas, C.; Nicolai, S. CrossTest: A cross-domain physical testbed environment for cybersecurity performance evaluations. In Proceedings of the 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), Stuttgart, Germany, 6–9 September 2022; pp. 1–8.
13. Lee, S.; Lee, S.; Yoo, H.; Kwon, S.; Shon, T. Design and implementation of cybersecurity testbed for industrial IoT systems. *J. Supercomput.* **2018**, *74*, 4506–4520. [CrossRef]
14. Jarmakiewicz, J.; Maślanka, K.; Parobczak, K. Development of cyber security testbed for critical infrastructure. In Proceedings of the International Conference on Military Communications and Information Systems, Cracow, Poland, 18–19 May 2015; pp. 1–10.
15. Wu, M.; Song, J.; Lucas Lin, L.W.; Aurelle, N.; Liu, Y.; Ding, B.; Song, Z.; Moon, Y.B. Establishment of intrusion detection testbed for CyberManufacturing systems. *Procedia Manuf.* **2018**, *26*, 1053–1064. [CrossRef]
16. Chan, E.Y.; Chan, H.; Chan, K.; Chan, P.; Chanson, S.T.; Cheung, M.; Chong, C.; Chow, K.; Hui, A.K.; Hui, L.C.K.; et al. Intrusion detection routers: Design, implementation and evaluation using an experimental testbed. *IEEE J. Sel. Areas Commun.* **2006**, *24*, 1889–1900. [CrossRef]

17. Vigna, G.; Robertson, W.; Balzarotti, D. Testing network-based intrusion detection signatures using mutant exploits. In Proceedings of the 11th ACM Conference on Computer and Communications Security, Washington, DC, USA, 25–29 October 2004; pp. 21–30.
18. Rossey, L.M.; Cunningham, R.K.; Fried, D.J.; Rabek, J.C.; Lippmann, R.P.; Haines, J.W.; Zissman, M.A. LARIAT: Lincoln adaptable real-time information assurance testbed. In Proceedings of the IEEE Aerospace Conference, Big Sky, MT, USA, 9–16 March 2002; Volume 6.
19. Singaraju, G.; Teo, L.; Zheng, Y. A testbed for quantitative assessment of intrusion detection systems using fuzzy logic. In Proceedings of the 2nd IEEE International Information Assurance Workshop, Charlotte, NC, USA, 8–9 April 2004; pp. 79–93.
20. Haines, J.W.; Goulet, S.A.; Durst, R.S.; Champion, T.G. LLsim: Network simulation for correlation and response testing. In Proceedings of the Systems, Man and Cybernetics Society, Information Assurance Workshop, West Point, NY, USA, 18–20 June 2003; pp. 243–250.
21. Zilberman, P.; Puzis, R.; Bruskin, S.; Shwarz, S.; Elovici, Y. Sok: A survey of open-source threat emulators. *arXiv* **2020**, arXiv:2003.01518.
22. Cheong, J. AutoTTP: Automated Tactics Techniques & Procedures 2019. Available online: https://github.com/jymcheong/AutoTTP (accessed on 26 September 2025).
23. BC-Security. Empire: Post-Exploitation Framework. Available online: https://github.com/BC-SECURITY/Empire (accessed on 26 September 2025).
24. Bader, L.; Serror, M.; Lamberts, O.; Sen, O.; van der Velde, D.; Hacker, I.; Filter, J.; Padilla, E.; Henze, M. Comprehensively Analyzing the Impact of Cyberattacks on Power Grids. In Proceedings of the 2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P), Delft, The Netherlands, 3–7 July 2023; pp. 1065–1081. [CrossRef]
25. Peuster, M.; Karl, H.; van Rossem, S. MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments. In Proceedings of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, 7–10 November 2016; pp. 148–153. [CrossRef]
26. Thurner, L.; Scheidler, A.; Schäfer, F.; Menke, J.H.; Dollichon, J.; Meier, F.; Meinecke, S.; Braun, M. Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems. *IEEE Trans. Power Syst.* **2018**, *33*, 6510–6521. [CrossRef]
27. *IEC 60870-5-104*; Telecontrol Equipment and Systems—Part 5-104: Transmission Protocols—Network Access for IEC 60870-5-101 Using Standard Transport Profiles. International Electrotechnical Commission: Geneva, Switzerland, 2006.
28. The MITRE Corporation. Adversarial Tactics, Techniques and Common Knowledge (ATT&CK). 2015. Available online: https://attack.mitre.org/ (accessed on 29 September 2025).
29. The MITRE Corporation. ATT&CK for Enterprise. 2015. Available online: https://attack.mitre.org/matrices/enterprise/ (accessed on 29 September 2025).
30. The MITRE Corporation. ATT&CK for Industrial Control Systems. 2020. Available online: https://attack.mitre.org/matrices/ics/ (accessed on 29 September 2025).
31. Codetta Raiteri, D.; Franceschinis, G.; Gribaudo, M. Defining formalisms and models in the Draw-Net Modelling System. In Proceedings of the International Workshop on Modelling of Objects, Components and Agents, Turku, Finland, 26 June 2006; pp. 123–144.
32. Marco, G.; Mazzocca, N.; Francesco, M.; Vittorini, V. Multisolution of complex performability models in the OsMoSys/DrawNET framework. In Proceedings of the Second International Conference on the Quantitative Evaluation of Systems (QEST'05), Torino, Italy, 19–22 September 2005; pp. 85–94. [CrossRef]
33. Calvarese, F.; Di Marco, A.; Malavolta, I. Towards a graphical representation for the Æmilia Architecture Description Language. In Proceedings of the 2nd Quantitative Information Workshop (infQ), Lipari, Italy, 27–29 June 2011.
34. Gilmore, S.; Gribaudo, M. Graphical modelling of process algebras with DrawNET. In Proceedings of the Tools presentation at the Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems, Urbana, IL, USA, 2–5 September 2003.
35. The MITRE Corporation. Cyber Analytics Repository (CAR). Available online: https://car.mitre.org/ (accessed on 29 September 2025).
36. Eaton, J.W. Octave. Available online: https://www.gnu.org/software/octave/ (accessed on 29 September 2025).
37. Rapid7. Metasploit Framework. Available online: https://www.metasploit.com (accessed on 26 September 2025).
38. Docker, Inc. Docker Stack CLI Reference. Available online: https://docs.docker.com/reference/cli/docker/stack/ (accessed on 26 September 2025).
39. Docker, Inc. Docker Swarm. Available online: https://docs.docker.com/engine/swarm/ (accessed on 26 September 2025).
40. Cerotti, D.; Codetta Raiteri, D.; Egidi, L.; Franceschinis, G.; Portinale, L.; Dondossola, G.; Terruggia, R. Analysis and Detection of Cyber Attack Processes targeting Smart Grids. In Proceedings of the IEEE-PES Innovative Smart Grid Technologies Europe (ISGT-Europe), Bucharest, Romania, 29 September–2 October 2019; pp. 1–5. [CrossRef]

41. Cerotti, D.; Codetta Raiteri, D.; Dondossola, G.; Egidi, L.; Franceschinis, G.; Portinale, L.; Terruggia, R. A Bayesian Network Approach for the Interpretation of Cyber Attacks to Power Systems. In Proceedings of the 3rd Italian Conference on Cyber Security, Pisa, Italy, 13–15 February 2019; Degano, P., Zunino, R., Eds.; Volume 2315, CEUR Workshop Proceedings.
42. Sabur, A.; Chowdhary, A.; Huang, D.; Alshamrani, A. Toward scalable graph-based security analysis for cloud networks. *Computer Networks* **2022**, *206*, 108795. [CrossRef]
43. Kahn, A.B. Topological sorting of large networks. *Commun. ACM* **1962**, *5*, 558–562. [CrossRef]
44. Miller, M. Metasploit's Meterpreter. Last modified 26 December 2004. Available online: https://www.hick.org/code/skape/papers/meterpreter.pdf (accessed on 29 September 2025).
45. Rapid7. Meterpreter—Metasploit Documentation. Available online: https://docs.metasploit.com/docs/using-metasploit/advanced/meterpreter/meterpreter.html (accessed on 29 September 2025).
46. Junior, G.T.; Clara Araujo Gomes Da Silva, A.; Dos Santos, R.P.; Kassab, M.; Graciano Neto, V.V. Are SoIS the Majority of SoS? An Exploratory Investigation of Subtypes of Systems-of-Systems in the Literature. In Proceedings of the 2025 IEEE/ACM 13th International Workshop on Software Engineering for Systems-of-Systems and Software Ecosystems (SESoS), Ottawa, ON, Canada, 3 May 2025; pp. 9–16. [CrossRef]
47. Linnosmaa, J.; Alanen, J.; Karadeniz, S.; Tiusanen, R.; Berger, J.; Malm, T.; Viitanen, K. *System of Systems Modelling for Safety and Cyber Security Assessments*; Number VTT-R-00555-24 in VTT Research Report; VTT Technical Research Centre of Finland: Espoo, Finland, 2024.
48. Fontes, R.R.; Afzal, S.; Brito, S.H.B.; Santos, M.A.S.; Rothenberg, C.E. Mininet-WiFi: Emulating software-defined wireless networks. In Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain, 13–15 November 2015; pp. 384–389. [CrossRef]
49. Guo, R.; Chen, Q.; Tong, S.; Liu, H. Knowledge-Aided Generative Adversarial Network: A Transfer Gradient-Less Adversarial Attack for Deep Learning-Based Soft Sensors. In Proceedings of the 2024 14th Asian Control Conference (ASCC), Dalian, China, 5–8 July 2024; pp. 1254–1259.