An Approach to Abstract Multi-stage Cyberattack Data Generation for ML-Based IDS in Smart Grids

Ömer Sen*, Philipp Malskorn*, Simon Glomb*, Immanuel Hacker*, Martin Henze†‡, Andreas Ulbig*
*IAEW, RWTH Aachen Univserity, Aachen, Germany
Email: {o.sen, i.hacker, a.ulbig}@iaew.rwth-aachen.de, {philipp.malskorn, simon.glomb}@rwth-aachen.de
†Security and Privacy in Industrial Cooperation, RWTH Aachen University, Aachen, Germany
‡Cyber Analysis & Defense, Fraunhofer FKIE, Wachtberg, Germany
Email: henze@cs.rwth-aachen.de

***Abstract*—Power grids are becoming more digitized, resulting in new opportunities for the grid operation but also new challenges, such as new threats from the cyber-domain. To address these challenges, cybersecurity solutions are being considered in the form of preventive, detective, and reactive measures. Machine learning-based intrusion detection systems are used as part of detection efforts to detect and defend against cyberattacks. However, training and testing data for these systems are often not available or suitable for use in machine learning models for detecting multi-stage cyberattacks in smart grids. In this paper, we propose a method to generate synthetic data using a graph-based approach for training machine learning models in smart grids. We use an abstract form of multi-stage cyberattacks defined via graph formulations and simulate the propagation behavior of attacks in the network. Within the selected scenarios, we observed promising results, but a larger number of scenarios need to be studied to draw a more informed conclusion about the suitability of synthesized data.**

***Index Terms*—Intrusion Detection, Smart Grid, Cyberattacks, Machine Learning, Knowledge Graphs**

I. Introduction

The cyber-physical characteristic of the power grid, in particular the increasing penetration of information and communication technology (ICT), addresses the issues of the distribution grid with respect to structural changes resulting from the integration of distributed energy resource (DER) [1]. In particular, it enables active grid operation at the distribution grid level and provides the backbone for the realization of smart grid (SG) concepts [2], [3]. This circumstance provides not only new opportunities, but also new threats resulting from the increasing interconnectedness of systems and actors [4]. The new threat landscape consists not only of threats arising from cyber domain interdependencies, but also of new attack surfaces for cyberattacks that threaten the stability and reliability of the grid [5]. To adequately address these new threats, defense and countermeasures must be integrated as an essential part of the SG infrastructure, encompassing a holistic approach of preventive, detective, and reactive measures [6]. One challenge in integrating countermeasures in the form of active security concepts into power grids is the consideration of legacy compliance, especially for countermeasures that actively interfere with grid operations [7]. More passive countermeasures such as intrusion detection system (IDS) offer the opportunity to support cybersecurity observability without imposing restrictive requirements on existing infrastructures, such as the performance specification of endpoint hosts [8]. More autonomous IDS build on Machine Learning (ML) that uses training data from operational or attack scenarios to classify anomalies or false positives from other IDS to support Security Operation Center (SOC) [9]. However, this requires attack data for the development and validation of these measures, especially data-driven approaches, which often cannot be accessed for security or privacy reasons [10]. To address these challenges, attack data are generated synthetically [11] or under laboratory conditions [12]. Laboratory environments can provide accurate data, but are costly to set up and consequently inflexible and difficult to scale. Synthetically generated data are more flexible and transferable to other use cases due to their level of abstraction, but can also suffer in quality due to inaccurate modeling and simulation.

To provide a basis for studying the cybersecurity of the SG by enabling the generation of attack data, a viable approach that compromises accuracy and effort is required. More specifically, the challenges we address in this paper are:

(i) Replication of the SG with all layers relevant for attack data synthesis and considering multi-stage patterns.
(ii) Execution of multi-stage attacks that replicate consistent and flexible attack patterns for dataset diversity.
(iii) Generation of adequate data sets using processable formats and consistent data structures based on use cases.

Therefore, in this paper, we propose a multi-stage approach to synthesize cyberattack data that allows generating datasets for ML based IDS in an abstract manner. To this end, we present an approach based on a knowledge graph that abstracts the network and multi-stage attack procedures in SG application scenarios. In particular, our contributions are:

1) We present the state of the art in synthesizing multistage cyberattack data and highlight the challenges of attack data generation (Section III).
2) We describe the overall approach consisting of modeling, knowledge graph representation, and simulation of multi-stage cyberattacks in SG (Section IV).
3) We demonstrate and discuss the dataset generation capabilities and quality of our proposed approach through case studies and comparisons with real datasets (Section V).

Author's version of a paper accepted for publication in Proceedings of 2023 IEEE Belgrade PowerTech.

II. Background

In this section, we describe the structure of SGs (Section II-A), anomaly detection (Section II-B), multi-stage cyberattacks (Section II-C) and attack data synthesis (Section II-D).

A. Smart Grid

Based on Purdue Enterprise Reference Architecture (PERA) for Industrial Control System (ICS), SGs are composed of five main levels: field devices, control systems, and management/operation as well as corporate systems [13]–[15]. Field devices, Level 0, make up the physical infrastructure of the SG and are responsible for monitoring and controlling various aspects of the SG. Control systems, Level 1, process and analyze the data collected from the field devices to make decisions and perform control actions. Management systems, Level 2, provide higher-level oversight and coordination of the SG and provide a connection to external networks and systems. The third level, Level 3, handles the management of production processes, such as managing batches, utilizing manufacturing operations management systems and manufacturing execution systems, and maintaining records of data. Level 4 encompasses systems like enterprise resource planning software, databases, email servers, and other tools that are used for logistics and communication, as well as for data storage. Lastly, the fifth level, Level 5, is the enterprise network, which is not part of the ICS but is used to gather data from the ICS systems to make business decisions.

B. Anomaly Detection

SGs, which are a type of ICS, are vulnerable to cyberattacks. Therefore, it is important to develop methods for detecting anomalies in network traffic that may indicate the presence of a cyberattack [16]. One approach to anomaly detection in SGs is to analyze network traffic patterns [17]. By analyzing sensor readings, control signals, communication logs, and network traffic it is possible to identify deviations from normal behavior that may indicate the presence of a cyberattack [18]. Another aspect to consider in anomaly detection for SGs is the possibility of multi-stage cyberattacks [19]. These types of attacks involve multiple steps, each of which may be difficult to detect individually. Therefore, it is important to consider not only individual anomalies, but also the potential for multiple attacks to be connected in a coordinated fashion. Overall, anomaly detection in SGs requires a combination of techniques that can analyze network traffic patterns and identify deviations from normal behavior. By using ML algorithms and other advanced techniques, it is possible to detect anomalies that may indicate the presence of a multi-stage cyberattack and take appropriate action to prevent damage to the system [20].

C. Multi-Stage Cyberattack

Multi-stage cyberattacks on SGs can be difficult to detect and mitigate due to their complexity and the ability for attackers to adapt and evolve their tactics [21]. To better understand and combat these types of attacks, researchers have developed various methods for structuring and analyzing multi-stage cyberattacks in a systematic and structured way. One common approach is the use of kill-chain like concepts, which divide the attack process into distinct stages such as reconnaissance, weaponization, delivery, and exploitation [22]. Another approach is the use of open-source datasets, such as the MITRE ATT&CK Matrix, which catalogs observed cyberattack incidents and can be used to identify patterns and trends in the attack process [23], [24]. The MITRE ATT&CK Matrix for ICSs is a comprehensive tool used to catalog observed cyberattack incidents in the ICS environment. It maps 96 types of attacks to 11 types of tactics and 67 new techniques specific to the ICS environment. The matrix is useful resource for generating synthetic multi-stage cyberattack data for ML-based anomaly detection and help security professionals understand and detect the various methods and tactics used by adversaries in attacks on ICSs.

D. Attack Data Generation

The generation of an attack dataset for anomaly detection in ICSs is a critical aspect of security research. One of the major challenges in generating multi-stage cyberattack data for ML-based anomaly detection is the reproducibility of the attack sequence [25]. It is essential to design an attack sequence that is suitable for the purpose of the dataset and implement it through automation using an integrated representation. This allows for the reproduction of the attack sequence and enables the explanation of the abnormal dataset information [26]. Additionally, generating a diverse range of attack sequences with specific requirements can be difficult, so a model that can reflect more diverse attack sequences is needed. One approach for generating an attack dataset is to analyze existing attack cases from the viewpoint of the MITRE ATT&CK framework, which catalogs observed cyberattack incidents and maps them to specific tactics and techniques. Another approach is to use an integrated representation, such as the hidden Markov model (HMM), to reproduce a sequence of attacks in a systematic and structured way. This allows for the creation of diverse attack sequences that reflect the reality of observed attacks and meet specific requirements, such as reproducibility and diversity.

III. Related Work & Problem Analysis

In this section, we present related work (Section III-A) and give a problem statement for this work (Section III-B).

A. Related Work

One direction within this research field are approaches that generate attack data in the lab. An example of this is the work of Sharafaldin et al. [27]. A dataset called CICIDS2017 was generated via a lab experiment. This contains recent attacks from DoS, DDoS, brute force, XSS, SQL injection, infiltration, port scan, and botnet. Another interesting approach to generating new datasets is presented by Cordero et al. [28]. Cordero et al. have developed the ID2T framework, which generates reproducible datasets for ML based IDS. Instead of generating more datasets, Pandey et al. [29] attempt to modify existing datasets. To test this method, the dataset UNSW-NB15

is considered. This consists of normal traffic and a variety of attacks from nine categories. With the Deep Auto Encoder (DAE) and the Wasserstein Generative Adversarial Network (WCGAN), additional data are generated and added to the original dataset. The main goal of this work is to investigate the detection quality of IDS trained on a synthetic dataset. For this purpose, we select a suitable ML based IDS method that exploits the structure and temporal dynamics of the dataset. For instance, Gwon et al. [30] investigate whether Long Short-Term Memory (LSTM) networks are suitable for binary classification of network packets. The LSTM network classifies each network packet as part of normal or as part of malicious traffic. Oliveira et al. [31] investigate the performance of different ML models for anomaly detection. In this context, the performance of a Random Forest, a MultiLayer Perceptron (MLP), and an LSTM network on the CIDDS-001 dataset are compared. The results show that the LSTM network performs with an accuracy of 99.94% and a F1 score of 0.91 provides the best results.

B. Problem Analysis

The process of creating datasets in a laboratory environment involves planning and implementing the infrastructure and scenarios, monitoring the system during execution, recording data, and ensuring proper execution. This process is highly labor-intensive and requires expertise in various subfields of computer science such as networks and cybersecurity. However, these datasets are not modifiable, making it difficult to add or remove attacks or change network topologies without re-running the experiment. Additionally, attack data is often underrepresented in these datasets as normal network traffic usually dominates. Care must be taken to prevent attacks on the network while data is being recorded to avoid compromising the dataset. This becomes more difficult to ensure depending on the size and complexity of the observed network. In addition, a separate network must be set up in the laboratory to record normal traffic, or a procedure must be implemented to obscure identifying information in the data for privacy reasons. There is also the problem of limited diversity in using public datasets to create additional datasets due to their limited availability. Researchers in ML also face the challenge of evaluating the performance of their models using only one dataset, leading to models that may not perform well in real network environments. This is further compounded by the lack of suitable datasets for training ML models.

IV. Abstract Multi-Staged Attack Data Synthesis

To overcome the problems mentioned in Section III-B, this paper presents an approach that abstracts multi staged attack data and generates it synthetically. To achieve this goal, an abstracted data format based on real data must first be devised. In the second step, a framework for synthetic generation of this data must be developed. For this work, different forms of data representation for the attack data are examined. Among them are network packets, network flows and IDS alert logs. The decision which of these data representations are most suited for the abstracted data synthesis is based on the following criteria:

(D1) Representation of data that is accessible in ICS and enterprise networks.
(D2) Representation of data that is collected in a standardized format.
(D3) Representation of data that is accessible in a human-readable and understandable form.

The investigation of the different forms of representation of the attack data has shown that IDS-alert logs are suitable for abstraction and synthesis as they fulfill all requirements. For Requirement (D1), the survey *A SANS 2021 Survey: Operational Technology (OT)/ICS Cybersecurity* shows that 62% of grid operators use signature-based IDS [32]. Requirement (D2) is satisfied as there are standardized file formats for IDS-alert such as the unified2 format used by the Snort IDS [33]. The last Requirement (D3) is met because IDS-alerts are designed to provide information to grid operators such that they can initiate countermeasures.

A. Requirement Analysis

In this section, requirements for the data synthesis framework are identified. The requirements can be divided into functional and non-functional requirements. Functional requirements describe the desired functionalities of the system while non-functional requirements describe the boundary conditions and quality in which those functionalities must be provided. The non-functional requirements include:

(N1) The framework must not use any additional hardware or virtualization.
(N2) The framework must not use additional applications for data generation.
(N3) The framework must not use attack signatures from existing IDS.
(N4) The framework must not require existing attack datasets.

Requirements (N1) and (N2) ensure that the effort for the configuration and setup does not exceed the effort of lab attempts. This is necessary because data from laboratory tests are closer to reality and should therefore be preferred if the production effort is the same. Requirements (N3) and (N4) ensure that poor data quality of existing datasets or poor documentation of existing IDS signatures do not affect this work. The following functional requirements set request to the synthesis framework and the abstract data:

(F1) Attack data must be generated in the form of identified data representation, i.e. as IDS log files.
(F2) Attack data must contain False Positives (FPs).
(F3) Attack data must contain False Negatives (FNs).
(F4) Networks must be flexible and scalable, i.e. must be freely configurable.
(F5) Networks must be representative, i.e. contain Enterprise and ICS Devices.
(F6) Attacks must be diverse, i.e. be freely configurable.
(F7) Attacks must be multi-staged.

TABLE I
DATA FIELDS OF THE ABSTRACTED IDS ALERTS.

Field	Description
Source-IP	IP address of the attacker.
Target-IP	IP address of the target.
Source-Platform	MITRE platform of the attacker.
Target-Platform	MITRE platform of the target.
MITRE-Tactic	Detected MITRE tactic.
MITRE-Technique	Detected MITRE technique.
Sensor-IP	IP address of the detecting sensor.
FP-Flag	A flag that indicates if the alert is FP.
Attack-Label	The attack label indicates which multilevel attack generated the alert.

The framework should generate data in the form of abstracted IDS alert logs, taking into account the different probabilities of FNs and FPs in real-world scenarios. Users should have the capability to configure networks and attacks for defining flexible and scalable scenarios. To provide a representative depiction of SG, the networks should include devices from both enterprise and ICS layers. The framework should be able to simulate both FNs and FPs cases based on IDS-detection accuracy and placement in network. Multi-stage attacks are more complex and have multiple phases or stages, rather than being a single action. To simulate different scenarios, the configurable attacks should be able to represent and simulate complex and diverse attack strategies and actions. This way, the framework will be able to provide a realistic representation of how IDS behaves.

B. Data Abstraction

In this section, the data format selected for synthesis is reduced to a simplified representation. In the following, the individual data fields of Snort's unified2 format are used as a basis because it is efficient and can store large amounts of data. MITRE ATT&CK matrix is used to abstract the cyberattack data and focus on essential information for cyberattack analysis, such as tactics and techniques used by attackers. This form of abstraction provides an efficient way to store and analyze large amounts of IDS data and allows for rapid identification of patterns and trends in the data. Table I gives an overview of the data fields of the abstracted data format.

C. Framework Overview

The framework architecture is designed to provide an overview of the system for data synthesis. It is implemented in Python, which is a suitable programming language for prototyping due to its debugging capability and high abstraction. The system architecture is divided into different modules, as shown in Figure 1. It illustrates the inputs, outputs and interactions of the different modules. All inputs to the system are in Resource Description Framework (RDF) data in Turtle format. For handling RDF, the Python library *rdflib* [34] is used, which allows for reading, writing and querying RDF files using SPARQL. To use RDF in combination with Shapes Constraint Language (SHACL), another library called pyshacl [35] is used, which allows for testing RDF data against previously defined constraints. In the following the interaction of the individual modules is addressed. First of all, the Network Manager reads a network configuration, which contains the description of a network. Network-Manager then generates a network graph from this information and provides functions for path finding. Next, the network graph is handed over to the attack coordinator. This module reads the attack configuration, which defines a multi-staged attack. Based on this configuration, a set of attack instances is determined by executing the attack on the network graph. These attack instances, along with the network graph, are then passed to the simulator. The simulator uses this information to generate IDS alerts for selected attack instances by checking if the attack communication is detected by an IDS. In addition, FN and FP are generated. The rate of FN and FP can be controlled. Finally, the simulator bundles the generated alerts into a csv file. Due to the efficient use of the integrated SPARQL query language in rdf, the generation is done in a reasonable time frame.

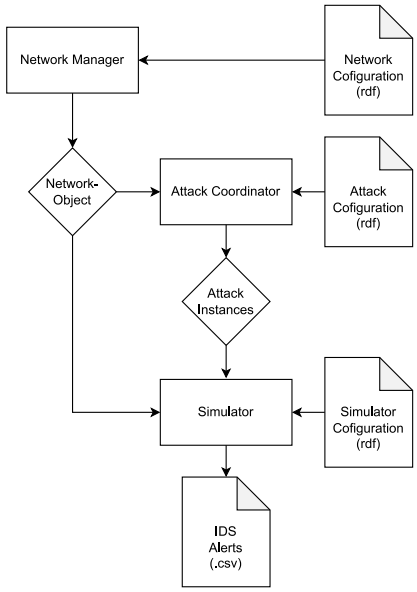

Fig. 1. Overview of the developed data synthesis framework.

D. Model Network

The network is modeled by a connected and undirected graph $G_{net} = (V_{net}, E_{net})$. In our model, an edge between two nodes $v_1, v_2 \in V_{net}$ exists if and only if there is a defined communication connection between them. The set of nodes V_{net} consists of four types of nodes, which can be used for constructing a network. These can be found in Table II. Each node must contain at least an IP address and a MITRE platform. The IP address must be unique, because in this framework it is used to identify nodes. MITRE platforms are a set of platforms that are referenced as possible targets by MITRE Techniques. These are later used to find possible targets for the configurable multi-stage attacks. For both attributes sets can be defined. The set *IP-addresses* can be defined with a Range of ipv4 or ipv6 addresses and the set *Platforms* is a subset of all

TABLE II
TYPES OF NODES IN THE NETWORK GRAPH.

Node Type	Description
Computer	Computers represent all devices in the network, which provide or use services for the control of the SG.
Router	Routers represent the edge router of the network.
Switches	Switches connect the individual components of the network and are possible nodes for the implementation of IDS.
Firewalls	Firewalls restrict the communication of the network and can be configured by rules.

MITRE Platforms, which can be selected based on the network being modeled. For instance, it can include IP addresses in the range of "172.32.0.0 - 172.47.255.255", "192.168.0.0 - 192.169.255.255" or "198.20.0.0 - 198.255.255.255".

$$\text{IP-addresses} = \{0.0.0.0, ..., 255.255.255.255\} \quad (1)$$

$$\text{Platforms} = \{\text{Network, Windows, Linux, ..., Containers, Control Server, Data Historian, ...,}\} \quad (2)$$

The Set of all nodes V_{net} consists of the union of all node sets $V_{net} = \{\text{Computer}\} \cup \{\text{Router}\} \cup \{\text{Switches}\} \cup \{\text{Firewalls}\}$. In the following the attributes of the individual node types are specified. Computers and routers contain only ip and platform therefore this can be formally stated as:

$$\text{Computer} = \{(p_i, ip_i) | p_i \in \text{Platform}, ip_i \in \text{IP-addresses}\} \quad (3)$$

$$\text{Router} = \{(p_j, ip_j) | p_j \in \text{Platform}, ip_j \in \text{IP-addresses}\} \quad (4)$$

Switches must also contain the information whether an IDS has been implemented. A Network-based IDS collects data from a switch via network taps or mirrored ports (SPAN-Ports) configured on the switch to detect and alert on suspicious network activity. For this an additional boolean variable *IsNIDSActiv* must be added. If the value is True, it is assumed that a IDS is active at this switch. The set of Switches is formally stated as follows:

$$\begin{aligned}\text{Switches} = \{&(p_k, ip_k, \text{IsNIDSActiv}) | \\ &p_k \in \text{Platform}, \\ &ip_k \in \text{IP-addresses}, \\ &\text{IsNIDSActiv} \in \{True, False\}\}\end{aligned} \quad (5)$$

Finally, the set of firewalls and their rules must be defined. The rules consist of a tuple with three items, which refers to a switch s_1, a switch s_2 and a computer c_1.

$$\begin{aligned}\text{Firewalls} = \{&(p_l, ip_l, R) | \\ &p_l \in \text{Platform}, \\ &ip_l \in \text{IP-addresses}, \\ &R \subseteq \text{Rules}\}\end{aligned} \quad (6)$$

$$\begin{aligned}\text{Rules} = \{&(c_i, s_{k1}, s_{k2}) | \\ &c_i \in \text{Computer}, \\ &s_{k1}, s_{k2} \in \text{Switches}\}\end{aligned} \quad (7)$$

To understand how the rules work, the possible edges and the concept of communication for G_{net} has to be defined first. The edges E_{net} of the network graph G_{net} consists of a subset of the cross product of V_{net} with Switches.

$$E_{net} \subseteq \{(e_1, e_2) | e_1 \in V_{net}, e_2 \in \text{Switches}\} \quad (8)$$

This implies that each edge must be connected to at least one switch. A device v_1 can start to communicate with another device v_2 if there exists a path from v_1 to v_2, where each firewall allows the device v_1 to communicate. A possible paths can contain either no, one, or several firewalls. In case there is no firewall on the path, device v_1 can communicate with device v_2. In the two other cases, each firewall must contain a rule that allows communication. The evaluation of the firewall rules works the same way in both cases. Without loss of generality we can consider the following path:

$$v_1 \rightarrow ... \rightarrow s_1 \rightarrow f_1 \rightarrow s_2 \rightarrow ... \rightarrow v_2 \quad (9)$$

For v_1 to communicate with v_2, the firewall f_1 must contain the rule (s_1, s_2, v_1). Here s_1 references the switch that is on the path directly before the firewall, s_2 references the switch that is directly after the firewall and v_1 is the device which would like to start the communication. Over the given switches each rule gets a direction, this is reflected in the fact that the communication in the graph G_{net} is not symmetrical. In this way, the communication in the network can be precisely controlled.

An example of a simple network graph can be seen in Figure 2. This contains seven nodes, including three computers, two switches, a router and a firewall with a rule. This rule allows computer with IP 192.168.0.20 to communicate with computer 192.168.0.22.

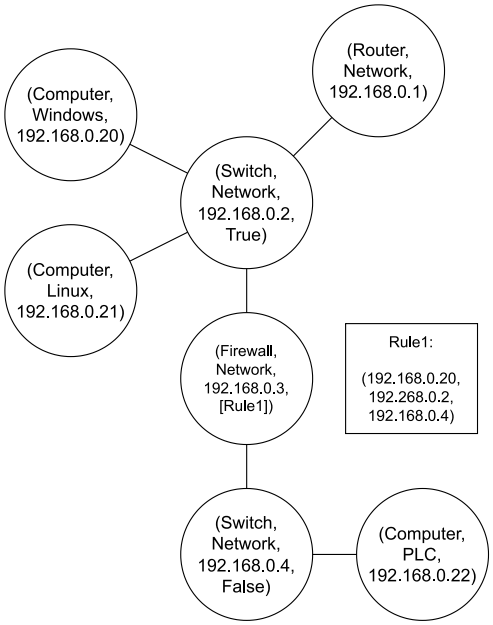

Fig. 2. An example network graph.

E. Model Attacks

Multi-staged attacks are represented as directed graphs $G_{att} = (V_{att}, E_{att})$. It should be noted that these are not attack graphs in a traditional sense. The set of nodes V_{att} represents wildcards, which later have to be filled with nodes from the network graph to execute an attack. V_{att} consists of a subset of the natural numbers:

$$V_{att} \subseteq \mathbf{N} \tag{10}$$

The edge set E_{att} represents the single step attacks that occur between the nodes and together represent the multi-staged attack. To model the attacks E_{att} the techniques and tactics of the MITRE ATT&CK Matrix are needed. Both are represented by a set containing the IDs.

$$\begin{aligned} \text{Tactics} = \{&\text{T0100, T0101, ..., T0111,} \\ &\text{T0001, T0002, ..., T0011,} \\ &\text{T0040, T0042, T0043}\} \end{aligned} \tag{11}$$

$$\begin{aligned} \text{Techniques} = \{&\text{T0800, T0801, ..., T0891,} \\ &\text{T1001, T1003, ..., T1649}\} \end{aligned} \tag{12}$$

With this information, E_{att} can be formally stated as:

$$\begin{aligned} E_{att} \subset \{&(v_1, v_2, step, tactic, technique) | \\ &v_1, v_2 \in V, \\ &step \in \mathbf{N}, \\ &tactic \in \text{Tactics}, \\ &technique \in \text{Technique}\} \end{aligned} \tag{13}$$

Each attack configuration also contains exactly one root node from which the attack originates. It additionally contains a single MITRE platform, which specifies the device type where the attack starts. For the remaining nodes, possible MITRE platforms can derived from the incoming edges. For this, the MITRE ATT&CK matrix is used again. It provides a mapping $TechniqueToPlatforms$ that assigns each MITRE technique to a subset of the MITRE platforms. This subset represents possible targets for the technique. If a node has only one incoming edge, the platforms can be found directly via $TechniqueToPlatforms$. If a node has several incoming edges, all platforms for the individual techniques must first be determined using $TechniqueToPlatforms$. After that the intersection of the different platform sets must be formed. The Platforms that are in the intersection are common goals of the different techniques. An example of an attack configuration G_{att} can be seen in the Figure 3.

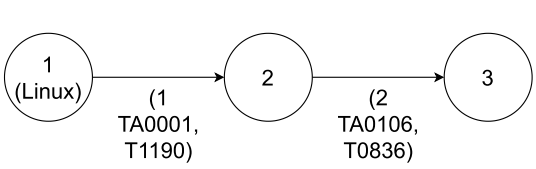

Fig. 3. A example attack graph.

F. Data Synthesis

The attack configurations discussed in Section IV-E are now used together with the network configurations from Section IV-D to generate appropriate IDS alerts. For this, the instances of the attack configuration for the network configuration must be found. An instance of an attack is represented by a mapping function, denoted as F, which assigns nodes from the set of attack nodes, V_{att}, to nodes in the network, V_{net}. For any pair of nodes, x_1 and x_2, in V_{att} that have an edge between them, denoted as Ex_1, x_2, the function F must map the nodes such that $F(x_1)$ and $F(x_2)$ are in V_{net} and $F(x_1)$ can communicate with $F(x_2)$, as outlined in Section IV-D. Additionally, for each node x_1 in V_{att}, the platform of the node $F(x_1)$ must be present in the platform set of node x_1, as outlined in Section IV-E. To find all instances, a complete graph traversal is performed using an algorithm described in Listing 1. The function $network\_graph.get\_step\_values(attack\_edges)$ is used to find a set of possible solutions for each attack step in E_{att} that comply with the conditions stated above. These sets are called $step\_values$. To generate instances from these partial solutions, the function $solve\_attack\_step$ is called for each attack step with the corresponding $step\_values$. The result is a list of dictionaries, where each dictionary represents the function F and thus represents an instance of an attack.

```
1
2   def solve_attack_step(attack_edge, step_values, instances):
3       new_instances = []
4       x1, x2 = attack_edge
5       for instance in instances:
6           for n1, n2 in step_values:
7               i_copy = instance.copy()
8
9               if i_copy[src] is None and i_copy[target] is None:
10                  i_copy[src] = n1
11                  i_copy[target] = n2
12                  new_instances.append(i_copy)
13
14              elif i_copy[src] == n1 and i_copy[target] == n2:
15                  new_instances.append(i_copy)
16
17              elif i_copy[src] is None:
18                  if i_copy[target] == n2:
19                      i_copy[src] = n1
20                      new_instances.append(i_copy)
21
22              elif i_copy[target] is None:
23                  if i_copy[src] == n1:
24                      i_copy[target] = n2
25                      new_instances.append(i_copy)
26      return new_instances
27
28  def find_all_instances(network_graph, attack_graph):
29      attack_edges = attack_graph.get_edges()
30      step_values = network_graph.get_step_values(attack_edges)
31      instances = []
32      for i, pair in enumerate(attack_edges):
33          instances = solve_attack_step(pair, step_values[i], instances)
34      return instances
```

Listing 1. Algorithm for finding all instances of an attack on a network.

Figure 4 shows an attack instance of the attack defined in Figure 3, executed on the network in Figure 2. To verify that this instance is valid, there are several conditions that must be met. Firstly, the root of the attack must be a node that uses the Linux MITRE platform, which is confirmed in this case. Secondly, the MITRE techniques used in the attack must match the platforms of the target nodes. In this instance, node 192.168.0.21 uses technique T1190 to attack node 192.168.0.20. T1190 is known to target Linux, Network, Windows and macOS platforms, thus the platform condition is fulfilled. In the second step, node 192.168.0.20 uses technique

TABLE III
THIS TABLE SHOWS THE ALERTS THAT WERE GENERATED WITH THE EXAMPLE INSTANCE IN FIGURE 4.

Name	Alert 1	Alert 2
Source-IP	192.168.0.21	192.168.0.20
Target-IP	192.168.0.20	192.168.0.22
Source-Plattform	Computer	Computer
Target-Plattform	Computer	Field Controller
MITRE-Taktik	TA0001	TA1190
MITRE-Technik	TA0106	TA0106
Sensor-IP	192.168.0.2	192.168.0.2
FP-FLAG	False	False
Attack-Label	Attack_1	Attack_1

T0836 to attack node 192.168.0.22, T0836 can target Control Server, Field Controller, HM and SIS, fulfilling this condition as well. Finally, the nodes must be able to communicate with each other. There is no firewall between nodes 192.168.0.21 and 192.168.0.20, they are connected to a switch, they can communicate with each other. However, on the path between nodes 192.168.0.20 and 192.168.0.22, there is a firewall which allows communication for node 192.168.0.20 on the path $20 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 22$ (IP addresses without Prefix 192.168.0.). Therefore, all conditions are met and this instance is valid. The next step is to convert the instances of the attack

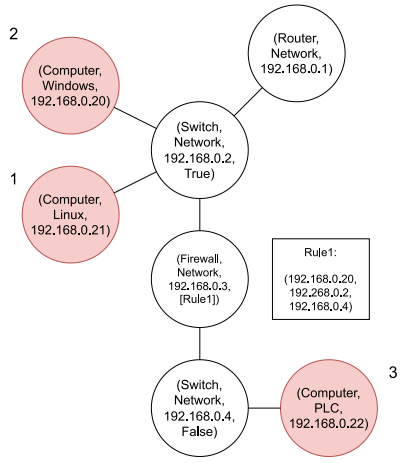

Fig. 4. A instance of the Attack from Figure 2 executed on the network graph from Figure 3.

into IDS alerts. This process requires the mapping function F, and the network graph G_{net} and the attack graph G_{att}. For each attack edge $(x_1, x_2) \in E_{att}$, it is verified if the path between the nodes $F(x_1)$ and $F(x_2)$ in the network graph contains a switch that is configured as IDS. If such a switch exists, an alert is generated with the attributes listed in Table I. The IP addresses and MITRE platforms are obtained from the network graph G_{net}, while the MITRE tactic, technique and attack label are obtained from the corresponding edges in the attack graph G_{att}. These alerts are considered True Positives (TPs), thus the FP flag is set to False. The alerts generated for the instance in Figure 4 can be found in Table III. There are two methods for adding FNs to the generated IDS alerts. One method is to selectively omit certain IDS devices from the network, which will increase the rate of FNs since attacks will go undetected. Another method is to randomly remove TPs from the data, which allows for more precise control over the proportion of FNs. This second method will be used here. In addition to generating FNs and TPs, it is also necessary to generate FPs. To do so, two nodes, x_1, x_2, are randomly selected from the network graph G_{net}, and it is checked if x_1 can communicate with x_2, and if there is an IDS device on the communication path. If such nodes are found, a suitable attack is randomly selected from the MITRE ATT&CK matrix and an alert is generated with it.

V. EVALUATION & DISCUSSION

In this section, we evaluate the effectiveness of using synthesized data for ML-based IDS by testing the performance of MLP and LSTM models, which are trained with synthesized data. The models are tested on both synthesized data and a data set from a laboratory experiment, evaluating both alert classification and attack detection. Additionally, the models are tested with both the full set of alert attributes and a reduced set of attributes that exclude the MITRE Tactic and Technique.

A. Procedure for Investigation

The investigation will be carried out as follows. First, configurations for the data synthesis framework are created for various scenarios. Then IDS-alert data sets with the required properties (TP-rate FN-rate and FP-rate) are generated. Finally, MLP and LSTM models are trained on this data and the performance will be measured. For the LSTM network, 2 layers are used. The first is an LSTM layer with 300 units and a dropout of 0.2 units and a dropout of 0.2. This layer receives a two-dimensional input whose size is equal to the feature vector times the length of the sequence. Three layers are used for the MLP. These include an input layer, a hidden layer, and an output layer. The input layer has the size of the feature vector. For the hidden layer, a dense layer is always used in the MLP. This has a size of 200 and uses the ReLU function as activation function. To give an indication of the performance on real data, the ML models, which are trained on the synthesized data, getting tested with the data from a lab experiment. For this purpose, the Packet Capture (PCAP) file of the laboratory experiment must be manually transferred to the abstract data format (cf. Section IV-B).

B. Scenarios

The network configuration used in the study were developed using the PERA and the National Institute of Standards and Technology (NIST) publication *Guide to Industrial Control Systems Security*. The NIST publication contains recommendations for securing ICS networks and, in particular, structures for network architectures [36]. The multi-stage attack configurations used in the study were created using the ICS kill chain and the MITRE software database. The ICS kill chain describes cyberattacks that aim to destroy the ICS infrastructure or temporarily disrupt production. The MITRE software database divides software into tools and malware.

For each software, the MITRE techniques used are listed. In addition, each technique has an explanation of exactly what it is used for. The created configurations replicate attacks of known malware. Among them are Stuxnet, Industroyer, WannaCry and PLC-Blaster.

C. Data Set Creation

To generate data with the presented framework in a way that ML models can utilize them, larger datasets have to be generated. In this work, datasets were generated by producing IDS alerts for a specific network configuration with a set of different attacks. The number of alerts for an attack configuration can be chosen arbitrarily. In addition, attributes of the whole dataset, such as the ratio of TP to FP or TP to FN, can be controlled. This is helpful to generate the appropriate training datasets for the different applications of the ML models. In our case these are the alert classification and the attack detection.

D. PCAP Transformation

The data of the laboratory experiment contains the network traffic of a test network in which the WannaCry ransomware is spreading [37]. To make the network packets of a PCAP file usable for our ML models, they first have to be converted into the abstracted IDS alerts. To generate IDS alerts from a PCAP file, the IDS Snort is used. Snort is a rule-based IDS and can also apply rules to archived data in the form of PCAPs [33]. In this work the Emerging Threats ruleset is used [38]. By applying the ruleset, the PCAP file with 46654 network packets is converted into a log file with 7874 alerts. This list contains a lot of duplicate alerts. To filter them out, duplicates that follow each other directly in time are removed. This reduces the number of alerts to 5406. These generated alerts contain a timestamp, the rule ID, a description, the protocol and the IP addresses. The number of alerts must be further reduced to manually assign labels. For this purpose, the priorities specified in the definition of each rule are used. All alerts with a low priority are removed. After this step, 66 alerts remain, which were generated by seven different rules. For these 66 alerts we have to manually assign MITRE ATT&CK tactics and techniques, while the alerts generated by the same rule get the same tactics or techniques. Using the manually assigned MITRE information and the knowledge about the platforms of the devices in the PCAP, the 66 alerts are transferred into the abstracted alert format. The Sensor ID attribute is ignored because the PCAP file does not contain any information about sensors. The FP label is set to False and the Attack label is set to WannaCry, since the PCAP contains only the WannaCry attack. FPs are needed to complete the dataset. These are generated using the synthesis environment for the period of the WannaCry attack. The final data set has 10 times as many FP as TP. Thus, the data set consists of 66 TP and 660 FPs.

E. Result

The results in this section were generated through a thorough process of training and testing the ML algorithms multiple times. The consistency in the results indicates the robustness of the models and the lack of significant difference in multiple runs of the algorithm makes it unnecessary to include standard deviation in the graphs. Additionally, the models have not been fine-tuned to their full potential, indicating room for improvement. Therefore, these results can be considered as a strong starting point for further optimization and refinement. The first experiment tests the performance of the ML models for alert classification. Here, the ML models should recognize the TP and thus separate them from the FP to simplify the manual processing of IDS alerts by the operator of the SG. For this two datasets were generated with 0% FN and with a TP to FP ratio of 100%. Alerts for attacks were generated until 2000 alerts were collected for each attack mentioned in Section V-B. This means that the datasets contain a total of 8000 TP and 8000 FP, this results in a balanced dataset. The MLP and LSTM are trained on the first data set and tested on the second one. The results of this test can be shown as a roc curve because we are looking at a binary classification tested on a balanced dataset. This roc curve is shown in Figure 5. It was expected that the LSTM would perform better than the MLP, since the LSTM can correlate successive alerts and thus derive further insights. The MLP, on the other hand, can only look at individual alerts. The roc curve shows that the LSTM performs better on the whole data set as well as on the data set which only uses the attributes source-IP, target-IP, source-platform and target-platform. Next,

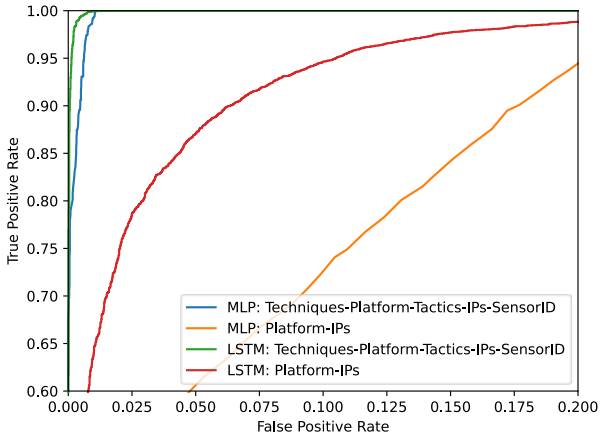

Fig. 5. Alert classification results on synthesized data. $x_{axis} \in \{0, 0.2\}$ and $y_{axis} \in \{0.6, 1.0\}$

the performance of the ML models trained on the synthesized data is tested on the laboratory data set from Section V-D. The results are visualized here as bar diagram of the TP rate and the FN rate. The roc curve cannot be used here because this is not a balanced data set. On the complete dataset the performance of the models is very good, reaching a TP rate of 0.99 and on the subset of the data set a TP rate over 0.8. The expected performance difference between the two models cannot be confirmed here. Here, the LSTM model no longer provides the expected advantage. The next experiment tests the

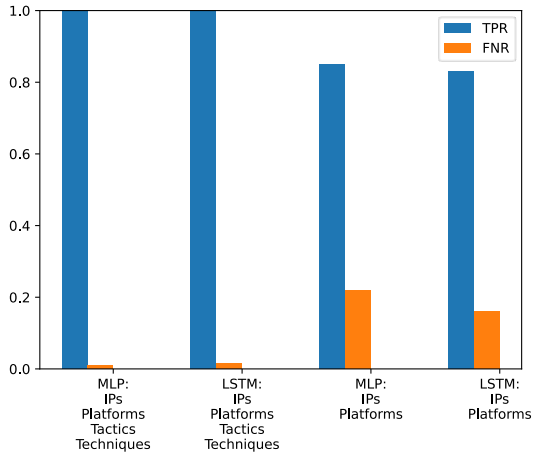

Fig. 6. Alert classification results on laboratory data.

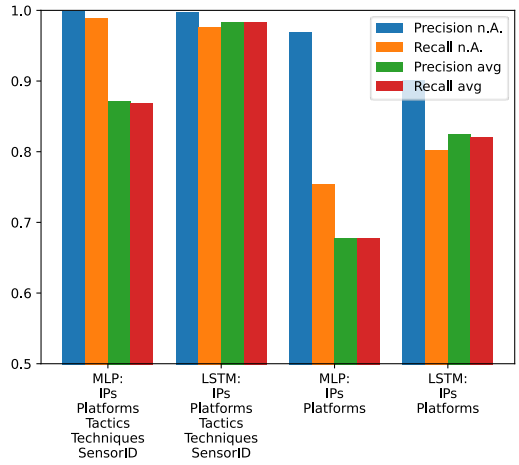

Fig. 7. Attack detection results on synthesized data (n.A = no Attack). $y_{axis} \in \{0.5, 1.0\}$

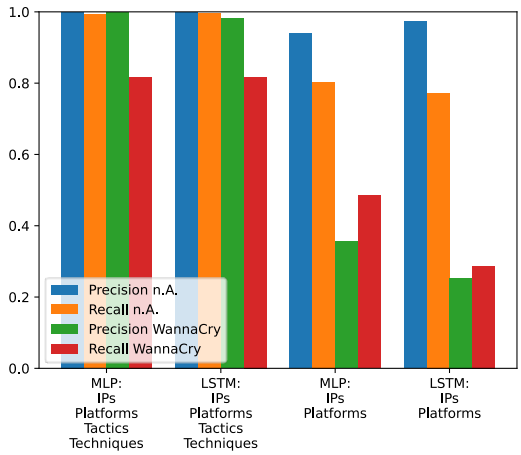

Fig. 8. Attack detection results on laboratory data (n.A = no Attack).

performance of the ML models for attack detection. Here, the ML models should recognize the ongoing Attack so that SG operators can take appropriate countermeasures. For this two data sets were generated with 0% FN and with a TP to FP ratio of 0.25. Again, alerts were generated until 2000 alerts were collected for each attack. This means that the data sets contain a total of 2000 for each attack and 2000 FP, so again balanced but over 5 classes. In this experiment, the ML models must assign the alerts to one of these 5 classes: Stuxnet, Industroyer, WannaCry and PLC-Blaster or False-Positive. The MLP and LSTM are trained on the first data set and tested on the second one. The results of this test can be shown as bar diagram, which takes the accuracy and recall into account. The class of false positives (n.A. = no attack) is considered individually, since there should be as few TP alerts as possible sorted into this class to not miss any attacks in the network. The results are shown in Figure 7. The results for the detection of FP (or no attack) for both models do not differ significantly, just as in the previous experiment. This can be seen from the fact that the Precision and the Recall for the class *no Attack* are in the same region. When detecting to which attack an alert belongs, alerts must be correlated to improve detection performance, since the same alert can belong to multiple attacks. This reveals the strength of LSTM. The average precision and recall across all classes is 10 percentage points higher with LSTM than with MLP. This holds for the whole data set as well as for the subset. Lastly, the performance of the ML models trained on synthesized data of the previous experiment is tested on the laboratory dataset. The results are shown in Figure 8. Again, the advantages of the LSTM are no longer apparent.

F. Discussion

In this section, the results of Section V-E are critically reviewed. Although the results of the ML models are promising, this is not an indication that proposed approach is transferable to real scenarios for the following reasons. The results of the LSTM were always better than those of the MLP in the experiments with the synthetic data. This indicates that the synthetic data contains a structure that can be recognized by the LSTM. The data from the laboratory experiment did not show such a clear performance advantage for the LSTM. This suggests that the attacks in the lab data do not have the same structure. This could be due to the manual steps necessary to convert the PCAP into a usable format. A second reason could be that in the data set the FP is so strongly overweighted. This could have made the examined sequence length of the LSTM too small. Another point that needs to be addressed is the performance of the ML models. The TP rate of the models is very close to 100%. This is usually a sign that the models are being used incorrectly or that the training data contains data that should not be there, such as labels. In our case, it is likely that the models can separate the TP and FP alerts too well via the MITRE tactic and technique attribute. In the attack configurations only a small subset of all techniques are used. This means that a large part of the FPs can be detected by their techniques, because they are exclusively used in the FPs. This leads to the next problem is the FP generation. The

FP from the synthesis environment are randomly generated. As already stated this is a problem because the attacks configured in the study does not cover all techniques. FP found in real environments are not random and can occur for many different reasons, such as bad configurations or hardware errors. Also, the framework and the ML models were not tested extensively. The reason for this is that all configurations in this work were configured manually. The results were generated with a set of 4 attack configurations and only one network configuration. To make a statement about whether the synthesized data can be used for ML based IDS, these test cases should be greatly expanded. Nevertheless, synthetic data has been shown to have many advantages. Among them are availability and the flexibility to generate data for different problems. Another advantage is that the data can be generated in a way that they are optimally suited for training ML models.

VI. Conclusion

To improve the security of SGs, ML-based IDSs are being implemented to detect attacks at an early stage. However, these systems require training data, which is often not available or suitable. This work investigated whether abstracted and synthetically generated data can be used to train ML-based IDSs. The investigation focused on IDS log files as a suitable data format for abstraction and synthesis. A system was presented that simulates multi-stage attacks on a network graph using the MITRE ATT&CK matrix, and generates IDS log files containing TPs, FPs, and FNs. The influence of the individual parameters of the abstracted data format on the performance of the ML models was investigated, and it was found that the MITRE ATT&CK related information in the IDS log files had the largest influence on their performance. The ML models were also tested on a dataset containing real attack data and were able to separate TP from FP and assign them to correct attacks. The resulting performance is promising in terms of training anomaly detection systems for SG without requiring a large number of datasets for training. Future work will address the study of synthetic, multi-stage cyberattacks in laboratory environments using digital twin approaches where the reference grid is replicated in our framework and a comparision with the lab is performed. In addition, other constellations of dataset construction will be investigated, e.g. the degree of blending of real and synthetic data also in terms of recognition and classification performance.

References

[1] Z. Vale *et al.*, "Distributed energy resources management with cyber-physical scada in the context of future smart grids," in *IEEE Mediterranean Electrotechnical Conference*, 2010.

[2] M. R. Hossain *et al.*, "Smart grid," in *Smart Grids*, 2013.

[3] D. van der Velde *et al.*, "Methods for Actors in the Electric Power System to Prevent, Detect and React to ICT Attacks and Failures," in *IEEE ENERGYCon*, 2020.

[4] Y. Yan *et al.*, "A survey on cyber security for smart grid communications," *IEEE communications surveys & tutorials*, 2012.

[5] C.-M. Mathas *et al.*, "Threat landscape for smart grid systems," in *ARES*, 2020.

[6] A. I. Kawoosa *et al.*, "A review of cyber securities in smart grid technology," in *ICCAKM*. IEEE, 2021.

[7] S. Sridhar *et al.*, "Cyber–physical system security for the electric power grid," *Proceedings of the IEEE*, 2011.

[8] T. Krause *et al.*, "Cybersecurity in Power Grids: Challenges and Opportunities," *Sensors*, 2021.

[9] K. Demertzis *et al.*, "The next generation cognitive security operations center: network flow forensics using cybersecurity intelligence," *Big Data and Cognitive Computing*, 2018.

[10] A. J. Burstein, "Toward a culture of cybersecurity research," *UC Berkeley Public Law Research Paper*, 2008.

[11] B. B. Zarpelão *et al.*, "How machine learning can support cyberattack detection in smart grids," in *Artificial Intelligence Techniques for a Scalable Energy Transition*, 2020.

[12] A. Ashok *et al.*, "Powercyber: A remotely accessible testbed for cyber physical security of the smart grid," in *IEEE ISGT*, 2016.

[13] M. J. Assante *et al.*, "The industrial control system cyber kill chain," *SANS Institute*, 2015.

[14] E. Hossain *et al.*, *Smart grid communications and networking*. Cambridge University Press, 2012.

[15] K. Es-Salhi, "Segmentation and segregation mechanisms and models to secure the integration of industrial control systems (ics) with corporate system," Ph.D. dissertation, 2019.

[16] M. Baptiste *et al.*, "Systematic and efficient anomaly detection framework using machine learning on public ics datasets," in *IEEE CSR*, 2021.

[17] Á. L. Perales Gómez *et al.*, "Madics: A methodology for anomaly detection in industrial control systems," *Symmetry*, 2020.

[18] K. Wolsing *et al.*, "IPAL: Breaking up Silos of Protocol-dependent and Domain-specific Industrial Intrusion Detection Systems," in *RAID*, 2022.

[19] Ö. Sen *et al.*, "On Using Contextual Correlation to Detect Multi-stage Cyber Attacks in Smart Grids," *Sustainable Energy, Grids and Networks*, vol. 32, 12 2022.

[20] D. Kus *et al.*, "A False Sense of Security? Revisiting the State of Machine Learning-Based Industrial Intrusion Detection," in *CPSS*, 2022.

[21] A. Ju *et al.*, "Mckc: a modified cyber kill chain model for cognitive apts analysis within enterprise multimedia network," *Multimedia Tools and Applications*, 2020.

[22] T. Yadav *et al.*, "Technical aspects of cyber kill chain," in *International symposium on security in computing and communication*, 2015.

[23] O. Alexander *et al.*, "Mitre att&ck for industrial control systems: Design and philosophy," *The MITRE Corporation*, 2020.

[24] W. Xiong *et al.*, "Cyber security threat modeling based on the mitre enterprise att&ck matrix," *Software and Systems Modeling*, 2022.

[25] S. Choi *et al.*, "A comparison of ics datasets for security research based on attack paths," in *International Conference on Critical Information Infrastructures Security*. Springer, 2019.

[26] S. Choi *et al.*, "Probabilistic attack sequence generation and execution based on mitre att&ck for ics datasets," in *Cyber Security Experimentation and Test Workshop*, 2021.

[27] I. Sharafaldin *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, 2018.

[28] C. G. Cordero *et al.*, "On generating network traffic datasets with synthetic attacks for intrusion detection," *ACM TOPS*, 2021.

[29] S. K. Pandey *et al.*, "Gan-based data generation approach for ids: Evaluation on decision tree," in *AISC: V14*, 2021.

[30] H. Gwon *et al.*, "Network intrusion detection based on lstm and feature embedding," *arXiv:1911.11552*, 2019.

[31] N. Oliveira *et al.*, "Intelligent cyber attack detection and classification for network-based intrusion detection systems," *Applied Sciences*, 2021.

[32] M. Bristow, "A sans 2021 survey: Ot/ics cybersecurity," *eng. In*, 2021.

[33] B. Caswell *et al.*, *Snort 2.1 intrusion detection*. Elsevier, 2004.

[34] M. A. Bamboat *et al.*, "Performance of rdf library of java, c# and python on large rdf models."

[35] J. E. Labra Gayo *et al.*, "Rdfshape: An rdf playground based on shapes," in *Proceedings of ISWC*, 2018.

[36] K. Stouffer *et al.*, "Guide to industrial control systems (ics) security," *NIST*, 2011.

[37] David Szili, "pcap of wannacry spreading using etnernalblue," 2017. [Online]. Available: https://www.malware-traffic-analysis.net/2017/05/18/index2.html

[38] Proofpoint Inc, "Emerging threats rules," 2022. [Online]. Available: https://rules.emergingthreats.net/