Nguyen, Tri; Ngoc Lam, An; Nguyen, Phu; Truong, Linh

Security Orchestration with Explainability for Digital Twins-based Smart Systems

Published in:
Proceedings - 2024 IEEE 48th Annual Computers, Software, and Applications Conference, COMPSAC 2024

DOI:
10.1109/COMPSAC61105.2024.00159

Published: 26/08/2024

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Nguyen, T., Ngoc Lam, A., Nguyen, P., & Truong, L. (2024). Security Orchestration with Explainability for Digital Twins-based Smart Systems. In H. Shahriar, H. Ohsaki, M. Sharmin, D. Towey, AKM. J. A. Majumder, Y. Hori, J.-J. Yang, M. Takemoto, N. Sakib, R. Banno, & S. I. Ahamed (Eds.), *Proceedings - 2024 IEEE 48th Annual Computers, Software, and Applications Conference, COMPSAC 2024* (pp. 1194-1203). IEEE. https://doi.org/10.1109/COMPSAC61105.2024.00159

Security Orchestration with Explainability for Digital Twins-based Smart Systems

Minh-Tri Nguyen
Department of Computer Science
Aalto University, Finland
tri.m.nguyen@aalto.fi

An Ngoc Lam
SINTEF
Oslo, Norway
an.lam@sintef.no

Phu Nguyen
SINTEF
Oslo, Norway
phu.nguyen@sintef.no

Hong-Linh Truong
Department of Computer Science
Aalto University, Finland
linh.truong@aalto.fi

***Abstract*—The Digital Twin (DT) paradigm has been largely adopted for many smart systems in various domains. Due to the heterogeneous and distributed nature of the physical twins, these systems increasingly incorporate disparate security tools, especially those based on service-based AI/ML capabilities. That presents numerous challenges in achieving a comprehensive understanding of security analytics and explainability in security operations carried out by ML-based security services, which require continuous monitoring and optimization to remain effective. This paper aims to support security service integration and automated analyses with enhanced explainability in DTs. We introduce a novel framework that unifies runtime contexts to facilitate security services unification and operation interpretation in security orchestration. We define a workflow and provide necessary services for generating security reports across physical and logical layers. Leveraging a centralized knowledge service, we let security analysts incorporate domain knowledge in automating incident reasoning and security enforcement at the logical layer. We demonstrate our explainability framework on a DT of an Industry 4.0 toy factory with two ML-based security services detecting network anomalies. Our experiments show a significant reduction in manual effort for orchestrating security incident analysis and mitigation.**

***Index Terms*—Security Orchestration, Digital Twin, Smart IoT Systems, Machine Learning, Explainability**

I. Introduction

Digital Twins (DTs) have made significant impacts on various industries such as manufacturing, healthcare, energy, transportation and logistics [1, 2]. With collaborative and synchronized operations across distributed and heterogeneous systems, their reliance on connectivity, real-time data, and integration with critical systems make them attractive to cyber attacks. Securing a DT-based smart system requires a holistic security solution to cover its diverse attack surfaces (e.g., physical devices, network infrastructures, access interfaces, authentication, and third-party integration). To date, this involves incorporating ML-based security solutions to coordinate and response to security incidents in real-time. That raises the demand for explainability in automated security orchestration, especially when integrating security technologies (e.g., frameworks, protocols, and standards) tailored for DT [2, 3]). On the other hand, existing security orchestration, automation and response (SOAR) approaches, as reviewed in [4], have not comprehensively leveraged the DT paradigm. While more and more SOAR approaches strive for automation, explainability remains an open research area.

The main question for this paper is that given many existing ML-based security tools and services for different security aspects in DTs, how can we (1) enhance the explainability of their outputs so that we can incorporate them appropriately, (2) facilitate explainable analyses for investigating root causes/problems and (3) provide explainability-enhanced security response/enforcement. Explainability is an essential requirement for achieving a comprehensive understanding of security orchestration. Such knowledge is crucial in continuous optimization, especially when using multiple ML-based security services in DTs. The explainability is shown in but not limited to the following abilities: (1) providing holistic insights into security operations/events in multiple security aspects; (2) interpreting and validating security operations/events without intricate technical details that facilitate collaboration between security teams, tools, and stakeholders; (3) enabling knowledge incorporation (e.g., incorporating domain-specific knowledge with knowledge of the system and multiple security contexts in automating security analyses and optimization).

Despite significant research [1], security orchestration, automation and response in the context of DTs still face the following challenges: (1) Diverse security tools from multiple vendors with different data structures pose numerous challenges in integrating/connecting information (e.g., system and security contexts) to build a unified runtime contextual knowledge. Security tools (e.g., Darktrace and Vectra AI) can detect security threats using AI/ML algorithms and provide actionable insights that aid in understanding and responding to security incidents effectively. However, their security analyses have not supported and incorporated knowledge of the physical systems, which is extremely complex and dynamic in DTs, thereby limiting their ability to analyze certain security events across DTs (e.g., cascading incidents). (2) Employing ML-based security services raises other challenges in continuous updating and re-training to adapt to evolving environments (e.g., data quality and patterns), which potentially generate computational burdens and lead to performance degradation if not managed effectively. These ML-based services mainly undergo fine-tuning and re-training based on observation and feedback from security analysts, consuming significant human effort. However, existing approaches have not yet leveraged the potential of the DT's contextual knowledge in evaluating and optimizing such tuning.

We are developing an explainability framework with the following contributions: (1) supporting continuous integration and data unification by creating a unified runtime context encompassing contextual knowledge of the DT and its ML-based security system; (2) introducing a workflow and essential services that utilize the unified runtime context to support automated security analyses with enhanced explainability. Our framework offers a comprehensive insight into security operations/events and enables security analysis across components/systems within DTs. It allows automated security analyses to operate at the logical layer without resolving the underlying systems and incorporate domain-specific knowledge and the runtime context in evaluating and optimizing performance for ML-based security systems.

The rest of this paper is organized as follows: Section II and Section III provide the background and introduce our explainability framework for supporting security orchestration. We present experiments and discussion in Section IV. Section V discusses the related work. We conclude the paper and outline the future work in Section VI.

II. Background

A. Security orchestration in Digital Twins

DTs can take various forms depending on applications and the complexity of physical systems [5]. In this study, we aim to support security orchestration for DTs whose physical entities are distributed across multiple systems/locations, as shown in Fig. 1. Such DTs can be found in, e.g., manufacturing [6], smart city [7], oil & gas industry [8] that are built atop interconnected systems, enabling, e.g., predictive maintenance and real-time operations optimization and decision-making, as discussed in [9]. In such DTs, physical entities are organized into different zones based on functionality, location, or security requirements. In each zone, physical entities are interconnected through a private network managed by a controller. Data from the physical entities is sent to the internet via multiple IoT gateways, which support different communication protocols such as MQTT, CoAP, and HTTP [10]. Within such DT-based smart systems, many common/ML-based security services are employed to address various security threats [1]. For example, Drarktrace and Vectra AI analyze network traffic; Ping Identity and Okta analyze user behavior for authentication and access control [11]. Then, security orchestration, which involves coordinating and managing security processes and tools, becomes highly complex and requires sophisticated solutions to effectively manage security policies, enforcement mechanisms, and incident response procedures across heterogeneous systems, protocols, and vendors [2].

B. Knowledge graph and its application in cybersecurity

Knowledge Graph is a structured data model for capturing the knowledge of objects, their attributes, and relations. It is extensively used across multiple domains that require knowledge representation and enrichment (e.g., Wikidata [12] and YAGO [13]). Knowledge graphs have been utilized to facilitate data integration and representation of DT models

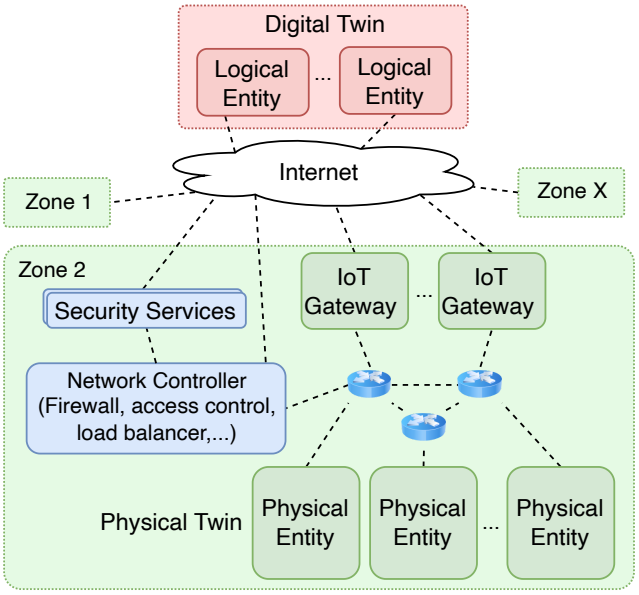

Fig. 1: Overview of a common deployment model for DT in manufacturing.

(e.g., SINDIT [14] and SEDIT [15]). In cybersecurity, knowledge graphs are used to model and represent the complex relationships between users, devices, applications, vulnerabilities, threats, and security controls [16]. Since knowledge graphs can provide contextualized views of complex security operations, an increasing number of security systems leverage knowledge graphs for multiple purposes, such as CSKG[1], SEPSES [17], and UCO[2]. The knowledge-based approaches also facilitate the development of proactive security measures, such as predictive analytics with AI/ML, automated response, continuous learning, and adaptation. That reduces human intervention in repetitive and error-prone tasks [16]. In this paper, we incorporate existing knowledge graphs from multiple security aspects and extend ML-specific attributes to improve explainability and address challenges in orchestrating ML-based security systems for DTs.

III. Explainability Framework for Orchestrating ML-based Security in DT

We are developing a framework to improve **R**untime e**X**plainability for **O**rchestrating **ML**-base **S**ecurity system (RXOMS) in DTs. RXOMS's objectives include: (1) Unifying knowledge of DT and its security system in multiple aspects within a centralized knowledge service to enable seamless integration of disparate security tools across DT environments (Section III-A). (2) Supporting automated security analyses, such as incident triage, investigation, and mitigation, to operate on the logical layer with reduced human intervention and errors, which ensures consistent and reliable execution of security operations in DTs (Section III-B).

[1]https://github.com/HoloLen/CyberSecurity_Knowledge_graph
[2]https://unifiedcyberontology.org/

TABLE I: Summarize the aspects of ML-based security services in Tool-specific Context for enhancing explainability.

	Aspects	Role in Explainability
Service	Configuration Settings	Explain the effectiveness dependency between how a security service is configured and its behaviors (e.g., thresholds, rules, and policies), supporting operation adjustment based on specific use cases or environments.
	Operational Parameters	Explain how a security service operates (e.g., scanning frequency, sampling rates, and data retention policies). They directly impact the reliability and compliance of security measures at runtime.
	Functionality Metadata	Explain the tasks and capabilities of the security service (e.g., anomaly detection, threat prevention). They help security analysts understand the core functions and how they contribute to overall security operations.
	Output Format and Content	Explain how a service presents its findings or results (format of alerts, reports, or logs). Analyzing the output content helps security analysts interpret the conclusions and integrate findings into broader security analysis.
ML-specific	ML Model	Explain the metadata of the embedded ML model (e.g., functionality, training data, feature extraction methods, and update mechanisms). They are essential for ensuring that ML-based security solutions remain explainable and trustworthy.
	Input Metadata	Explain types of input data, data sources, and data quality. That allows security analysts to assess the relevance and patterns of input data, which influence the performance of the security service.
	Performance & Accuracy	Explain the service performance with ML-specific metrics, such as detection confidence, false positive rates, and average accuracy. They provide quantitative measures of effectiveness, reliability, and compliance.

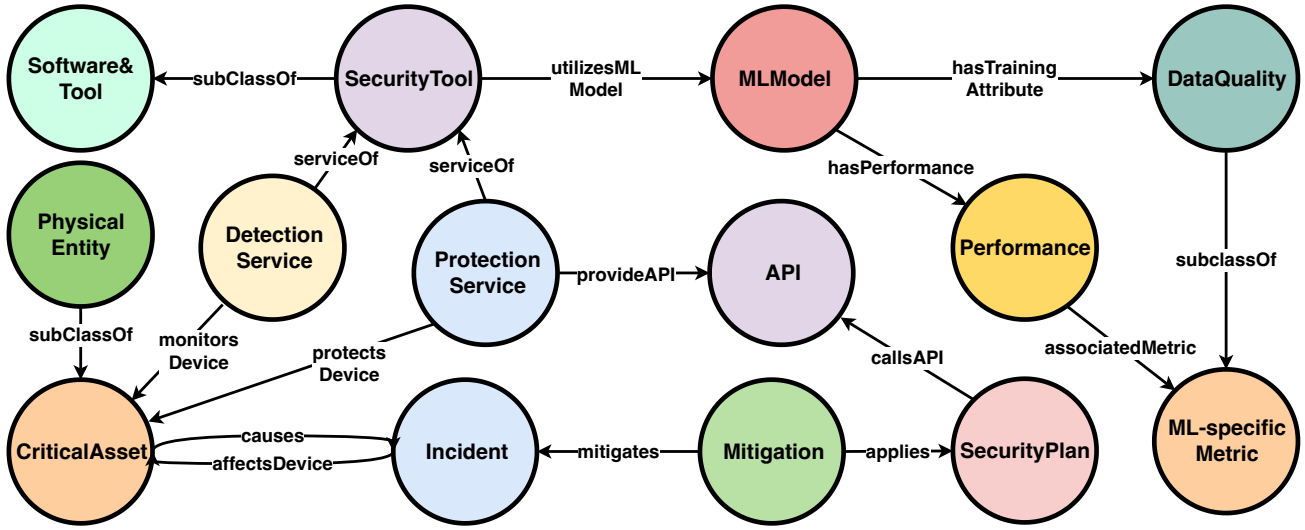

Fig. 2: A subgraph represents runtime knowledge consolidated from Environment, Tool-specific, and Security Context.

A. *Unifying contexts in Knowledge Service*

We view each entity in the DT as a subject in engineering analytics, so-called *analytic subject*. To understand the subject (e.g., security operations and events on the subject), we must examine its relevant information from suitable contexts at runtime. (1) *Tool-specific Context*: includes knowledge of the security service related to the subject at a certain runtime, such as functionality, performance, configuration, input source, and input quality. (2) *Environment Context*: provides knowledge of the associated entity within the DT, including the relationships with other entities, e.g., direct network connections to other entities or being operated by a specific user. (3) *Security Context*: provides knowledge of the associated entity within multiple security aspects, such as whether the entity is currently experiencing unauthorized access or has been detected with malware infections. To access such contexts in real-time, we develop a centralized knowledge service that incorporates the contextual knowledge from DT and its security systems as a unified *Runtime Context*.

1) Aspects of security services in Tool-specific Context: We assume that ML-based security services (of a security tool) operate as black boxes and we have limited control via configurations, operational parameters, and interfaces. Based on intensive analyses, we summarize a list of aspects for constructing the Tool-specific Context in Table I. Security services, especially ML-based services, must be configured appropriately to ensure they function correctly with expected outcomes. Their performance (e.g., accuracy and latency) heavily relies on the quality of the data they receive and how they process the data. Thus, all aspects affecting the performance (manufacturer configurations, ML models, and data quality) must be constantly updated, supporting security analysts to explain the *analytic subjects* at runtime. We use knowledge graphs to represent the complex and dynamic relationships among aspects in the Tool-specific Context. All security services share the same graph schema but do not necessarily possess all the aspects (in Table I). As more and more security services are integrated, new aspects from emerging services will be incorporated to enrich the schema.

2) Integration of environment, tool-specific, and security knowledge: We implement the Environment and Security Contexts as knowledge graphs based on prior works. The Environment Context is specific to individual DT, which has been developed by the DT's owner. For Security Context, we

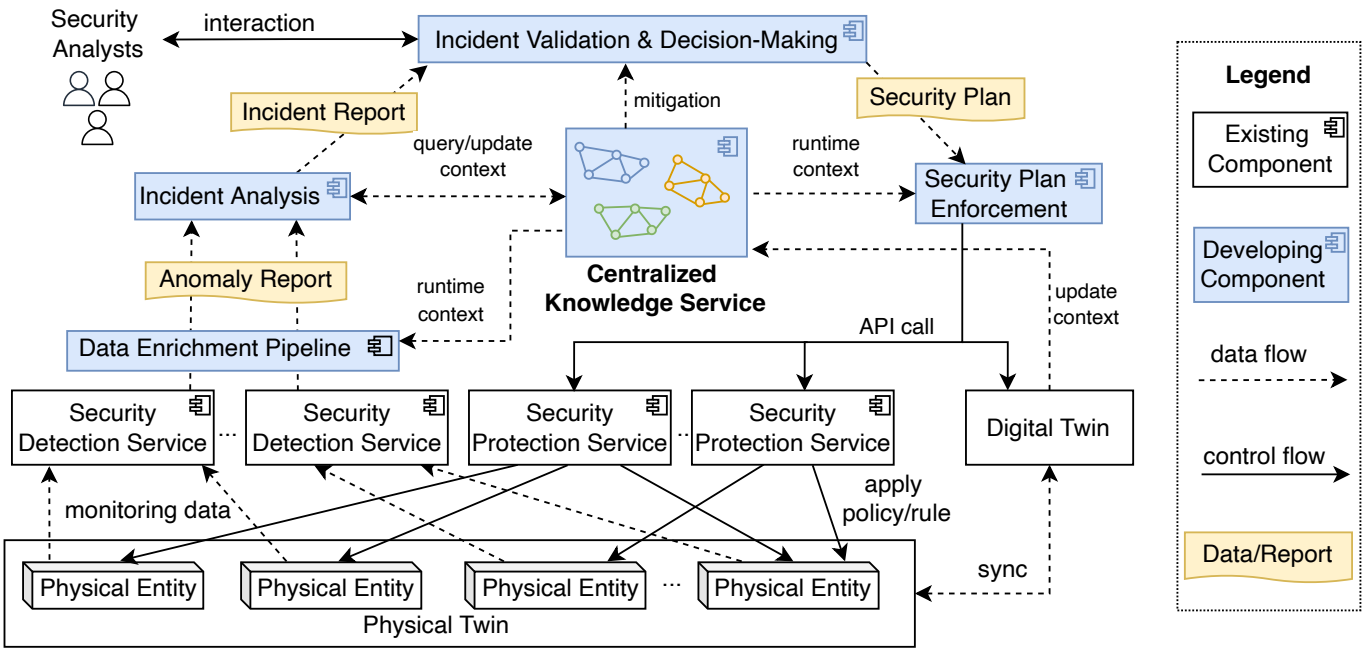

Fig. 3: Workflow and essential components in RXOMS for enhancing explainability in security orchestration and analysis.

build a graph that synthesizes existing knowledge graphs in CSKG and SEPSES for multiple security aspects (e.g., network, authentication, and physical security) and consolidates the objects and their attributes.

In our work, the environment, tool-specific, and security contextual knowledge provide comprehensive information for analyzing the *analytic subject* (demonstrated in Section IV). Integrating them into a unified knowledge service streamlines and simplifies the process of accessing, managing, and utilizing relevant knowledge in security analysis and orchestration, avoiding redundancy and inconsistencies, thereby enhancing the efficiency of security operations. In the unified knowledge graph, an object can represent a physical/logical entity in DTs, a class (e.g., anomaly and incident) in the Security Context, or a security tool/service. In different contexts, there are objects with a subset of similar attributes within existing knowledge graphs. As these objects show similarity across multiple contexts, we define a new object in the unified context that encapsulates all the common attributes. Then, those considered similar objects become subclasses of the newly created object. For example, the *SecurityTool* in the Environment Context is a subclass of *Software&Tool* in the unified Runtime Context, or a *PhysicalEntity* in the Environment Context is a subclass of *CriticalAsset* which is linked to the Security Context. This way, information of the *analytic subject* is linked to the Tool-specific Context (e.g., *ProtectionService* and *DetectionService*), and Security Contexts corresponding to multiple security aspects. Moreover, we define new objects to capture ML-specific attributes, such as *MLModel*, which encapsulate training aspects (e.g., training *DataQuality*) to enrich the information of ML-based security services; and *Performance* associated with *ML-specificMetric*(s) (e.g., average detection accuracy/confidence and false negative rate) to support quantitative explainability of effectiveness, reliability, and compliance. For graph storage and querying, we utilize Stardog[3], a state-of-the-art triplestore known for its commendable performance.

B. Utilizing Centralized Knowledge Service for orchestration

To leverage the *Centralized Knowledge Service* for enhancing explainability in security for DTs, RXOMS implements a workflow with essential services, as depicted in Fig. 3, demonstrating the important role of the Runtime Context in various processes throughout security analysis and orchestration.

1) Consolidating and enriching anomaly reports: The deployment and utilization of multiple security services for DTs lead to the diversity of anomaly reports. Discrepancies in data structure among different security reports hinder the ability to correlate knowledge in multiple security aspects and understand cross-zone/subsystem security problems. Thus, RXOMS employs a *Data Enrichment Pipeline* (DEP) to enrich the output of security (detection) services. The consolidated *anomalyReport* follows the schema depicted in Fig. 4. The report schema includes: *monitoringInfo* representing information of input data (e.g., data source and quality); *anomalyMetric*(s) representing outcomes of the security service (e.g., decisions and ML-specific metrics); *physicalEntityInfo* representing attributes of the entity from the Environment Context; *detectionInfo* representing multiple aspects of the security service in the Tool-specific Context, that include the current configuration and performance of the ML models. DEP, as a streaming

[3]https://www.stardog.com/

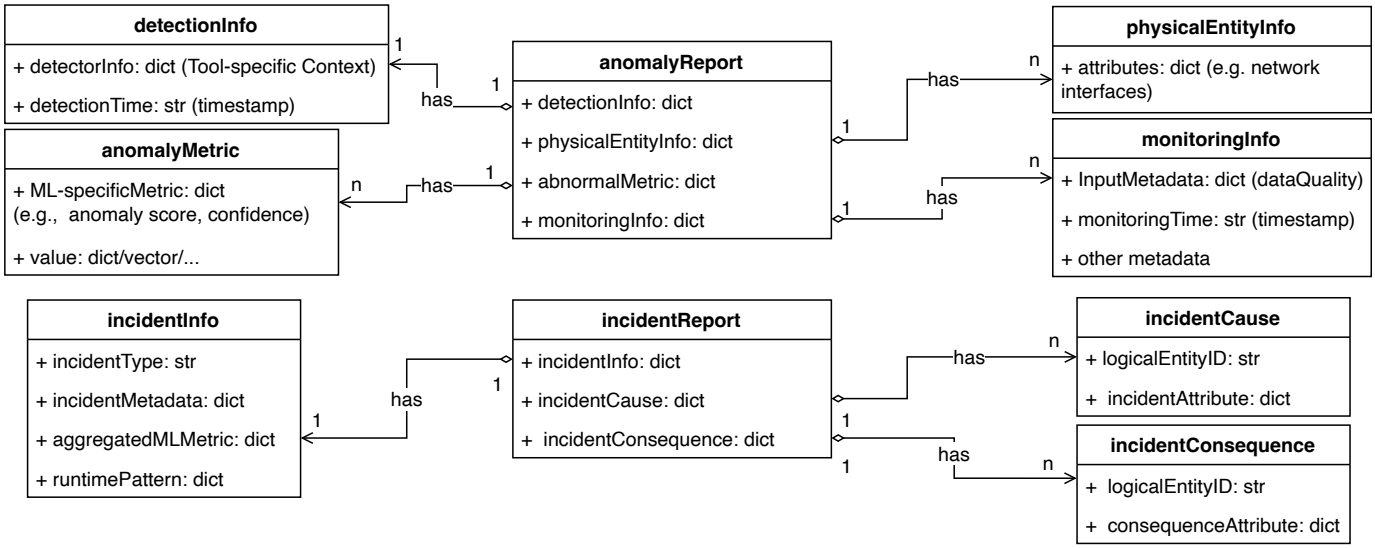

Fig. 4: Anomaly report and incident report schemas.

process, provides an API that allows security services to forward their output, including detected anomalies, metrics, attributes, and service identity. Based on that, the DEP queries Runtime Context from the *Knowledge Service* to enrich the anomaly report, which will be demonstrated in Section IV.

2) Supporting automated security analysis with enhanced explainability: The *Incident Analysis* service is responsible for analyzing the anomaly in its relation to other entities within the Environment Context and other anomalies in different security aspects within the Security Context. In RXOMS, the knowledge service provides multiple APIs for automated security analysis to query information related to the *analytic subjects*, such as its relationships with others, relevant security tools, and anomalies at runtime. Thereby, security analysts are able to incorporate domain-specific knowledge into their analyses to perform real-time incident identification and tracing. For handling an anomaly (e.g., a sudden increase in network traffic), an automated analysis is able to: (1) utilize the entity's relationships (e.g., network flow) to automatically trace the source(s) causing the anomaly, the destination(s)/endpoint(s) of the anomaly, and affected entities along the tracing path; (2) assess the pattern, severity, and impact of the anomaly on the related entities to identify the security incident (e.g., DDoS attack); (3) leverage the Security Context on the anomaly source(s) to determine the root cause in real-time (e.g., unauthorized user access, malware, or system fault); (4) examine the Runtime Context to determine whether this is a new or ongoing incident to update the Runtime Contexts correspondingly.

Once a new incident is detected, the *Incident Analysis* service generates an *incidentReport* following the schema shown in Fig. 4. The report includes: *incidentCause* representing the logical entity causing the incident (e.g., user, software, or device) and attributes describing its anomaly behavior; *incidentConsequence* representing related logical entities with their attributes describing the impact on individual entities; *incidentInfo* representing metadata that describes the type of incident according to MITRE and attributes detected at runtime (e.g., *runtimePattern*). Especially, it contains *aggregatedMLMetric*, which presents the aggregation of *ML-specificMetric*(s) from the analyzed *anomalyReport*(s). In other words, an incident is concluded based on multiple anomalies across the DT's systems, detected by several ML-based security services with different reliability (e.g., confidence and accuracy). For example, the *aggregatedMLMetric*, being the average/minimum confidence, demonstrates the certainty of a specific incident detection to support security analysts in validating and making decisions. We use Environment Context to link all physical objects to the corresponding logical objects to omit their technical details in the report. For example, instead of reporting an incident emerging from IP address *X*, we report the incident associated with the logical entity whose physical twin (physical entity) was attached to *X* at the time of the incident. That enhances explainability for incident analyses on dynamic underlying systems (e.g., physical entities reconnect frequently and use different IP addresses). The report is abstracted from technical details and only presents incident knowledge with objects at the logical layer, which will be illustrated in Section IV.

3) Enhancing explainability in incident validation and mitigation: Making decisions to handle an incident is a process that requires the validation of the security analysts. Our *Incident Validation & Decision-Making* (IVDM) service provides APIs allowing security analysts to query relevant information of the *analytic subject* (e.g., runtime metrics from security services and status of the entity at certain runtime). Thus, security analysts can verify the condition of all *analytic subjects* mentioned in the incident report. The service relies on the detected incident type to automatically query incident mitigation from the *Knowledge Service* to assist security analysts in making decisions. In the current implementation, we have just supported a basic incident-mitigation mapping

TABLE II: Sensor data collected from the Training Factory.

Physical Entity	Sensors	Gateway
Condition Sensor	14 Sensors for air quality, pressure, temperature, and camera positions	MQTT
		OPC-UA
Multi-Processing Station	2 Sensors for MPO status	MQTT
Sorting Line	2 Sensors for sorting line status	MQTT
Highbay Warehouse - HWB	6 Sensors for HWB status and positions	MQTT
		OPC-UA
Delivery And Pickup - DPS	4 Sensors for DPS status	MQTT
Robot-VGR	9 Sensors for VGR status and positions	MQTT
		OPC-UA

according to MITRE's public Cyber Threat Intelligence (CTI). Incident mitigation can be added by security analysts or based on historical incident solutions. Every mitigation as an abstract strategy is mapped into a list of actionable tasks/workflows known as a *Security Plan*. Since mapping incidents to mitigation and subsequently to security plans is a complex process, it will be addressed in our future studies. Then, the IVDM service uses the Environment and Tool-specific Contexts to map the tasks to specific API calls to corresponding security services. As a result, we obtain a security plan as a list of API calls applied to the specific logical objects. At this point, the responsibility of the *Security Plan Enforcement* service is to execute these API calls with the parameters defined in the Tool-specific Context. The values passed into these parameters (e.g., IP address, ports, and devices) are physical attributes of the associated objects in the Environment Context. Repetitive tasks such as querying Runtime Contexts, mapping objects and attributes, and calling APIs are automated to reduce human efforts and the risk of errors. To ensure the correctness of the Runtime Context, all changes in the physical system or digital twin must be synchronized and timely updated.

IV. Experiments

A. Experiment setup

In this study, we conduct experiments on SINDIT, a Digital Twin of the Fischertechnik Training Factory [18]. The factory includes 37 sensors belonging to six physical entities. The entities (physical twin) constantly send sensor data to the digital twin through MQTT and OPC-UA gateways. The list of entities, sensors, and gateways is described in Table II. The factory's network comprises two zones. Zone 1 (Z1) includes *Switch S1*, *MQTT Gateway*, *Multi-Processing Station*, *Sorting Line*, and *Delivery & Pickup*, which only send out data via the MQTT gateway. Zone 2 (Z2) includes *Switch S2*, *OPC-UA Gateway*, *Condition Sensor*, *Robot-VGR*, and *Highbay Warehouse*. Physical entities in Z2 communicate with MQTT and OPC-UA gateways so they can have cross-zone access.

We use Mininet [19] to emulate the factory network, as shown in Fig. 5. From the real dataset extracted from SINDIT, we estimate the data volume and number of network packets from each entity to individual gateways and vice versa. Based on that, we used a Python simulation program to generate the network traffic of 10 days scaled down within 30 minutes for the experiments. We employ an SDN controller developed in

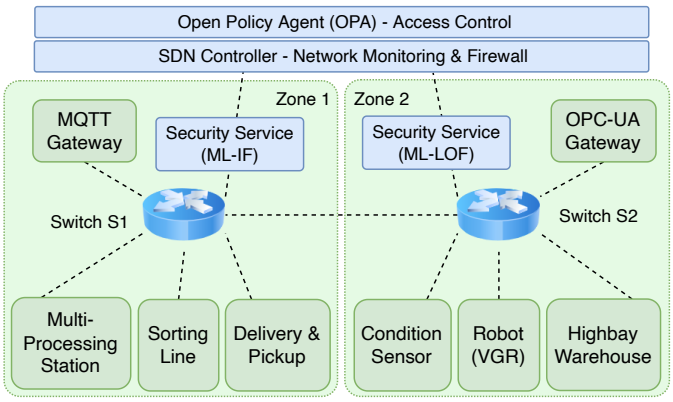

Fig. 5: SINDIT's Network Simulation.

the Ryu framework [20] to manage the network. The controller monitors and reports network stats collected from switches every second. The network stats on each switch include input port (*in_port*), ethernet destination MAC address (*eth_dst*), packet count (*packetCount*), and byte count (*byteCount*) on individual network flows. The physical network is managed by a firewall deployed on the SDN controller that offers common APIs to add/update/remove the firewall's rules. Besides, we employ an OPA [21] engine for logical access control from entities in SINDIT to the gateways.

We emulated two ML-based *Security Detection Services* to detect anomalies in the two zones separately (Fig. 5). These services embed Isolation Forest (ML-IF) and Local Outlier Factor (ML-LOF) respectively. These ML models are widely used in network traffic analysis [22, 23] and are chosen due to the lack of access to industrial ML-based Security Detection Services suitable for the factory. The ML-IF returns the output as a data frame with a pre-defined schema, including *networkFlow(switch, in_port, eth_dst)*, *prediction*, *confidence*, and *anomalyScore*. The ML-LOF returns a dictionary output with different ML-specific metrics, *networkFlow(switch, in_port, eth_dst)*, *prediction*, *nearestNeighbor*, and *distance*. These services detect anomalies in network traffic based on the time-series of *packetCount* and *byteCount* of all network flows in their zones. The ML models inside the services are trained on historical network data to detect anomalies in the current network traffic. Similar to real-world scenarios, we assumed that the models must be constantly re-trained to adapt to the evolving network patterns.

In our experiments, we simulated DDoS attacks by selecting physical entities as attackers and gateways as attack targets. We induce rapid surges in data packet transmission from attackers to targets with configurable network traffic (byte and packets) and time periods. The attacks resulted in anomalies detected by the ML-based security services. Based on the anomaly reports, the processes of tracing, identifying incidents, and generating security plans are automated. Security analysts only validate the incident reports and security plans to trigger the security plan enforcement.

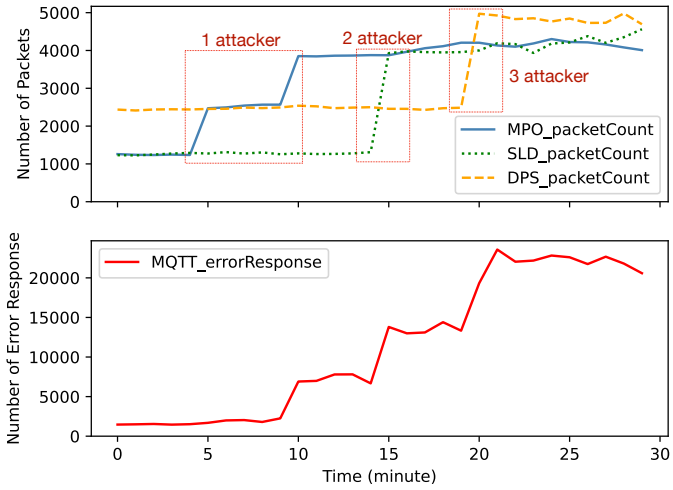

Fig. 6: Simulate attacks.

B. Experiment 1: Runtime explainability in identifying and mitigating security incident

In this experiment, we perform several attacks from the *Multi-Processing Station* - MPO, *Sorting Line* - SLD, and *Delivery and Pickup* - DPS to the *MQTT Gateway*. Fig. 6 shows the number of network packets generated by these entities and the number of error responses observed on the gateway. After the first 5 minutes, we increase the number of packets generated from MPO at a certain level without causing overflow on the *MQTT Gateway*. Every next 5 minutes, we increase network traffic and the number of attackers to cause disruptions on the *MQTT Gateway*[4].

Listing 1: Excerpt of an incident log from Vectra AI.

```
{
  "UTCTimeStart": "1701461667", "UTCTimeEnd": "1701461852",
  "dd_dst_ip": "10.0.0.4",
  "dd_proto": "TCP",
  "category": "DDoS",
  "certainty": 72,
  ...}
```

Without Runtime Context, security analysts only receive basic information about anomalies. As shown in Listing 1 (based on Vectra AI Syslog[5]), *certainty* represents the confidence level of the ML-based detection. Thus, it is difficult to determine whether the detected anomaly is reliable. For instance, if the security service is misconfigured, making mistakes in detection, or low-quality input data results in low detection quality. Listing 2 shows an excerpt of our anomaly report consolidated by a data enrichment pipeline. Based on the Runtime Context, we enrich the report with the following information: *detectionInfo* including the ML model metadata and security service configurations (e.g., threshold); *monitoringInfo* including the source and quality of data inputted into the ML-based detection (e.g., sample rate and consistency); *anomalyMetric* including the detection confidence, anomaly score, and the current anomaly value (*byteCount*); and *physicalEntityInfo* to support tracing in subsequent analyses.

To identify incidents by analyzing anomalies without a unified runtime context, security analysts must query information from various sources (logs and databases) based on relationships among entities, posing challenges in tracing and managing data consistency. Especially in dynamic environments like SINDIT, physical entities (factory assets) frequently crash and restart with new attributes. Thus, automating these analyses in real-time becomes extremely challenging. As long as all changes from physical systems and SINDIT are timely and accurately updated in the *Knowledge Service*, our *Incident Analysis* service allows analysts to configure tracing rules examining relationships (of related objects) from the runtime contexts and define rule-based algorithms to identify incidents with enhanced explainability and responsiveness.

Listing 2: Excerpt of *anomalyReport* when detecting abnormal network flows in a DDoS attack.

```
"reportID": "276852de-744a",
    "detectionInfo":{
        "detectionTime":"1701462137",
        "configuration": {"MLModel":{}, "threshold":{},...}
        ...},
    "physicalEntityInfo":{
        "0192eb5c-e33b": { # entityID
            "attribute": {
                "in_port": 2,
                "eth_dst": "10.0.0.4"
            },...}
    },
    "anomalyMetric":{
        "detectionConfidence": 0.72,
        "anomalyScore": -0.95,
        "byteCount": {
            "value": [486565],
            "unit": "Mbyte"}
    ...},
    "monitoringInfo":{
        "monitoringTime": "1701461667"},
        "dataQuality": {"consistency": 100,...}
        "sampleRate": 1,
    ...
```

Table III presents the number of interactions connecting to the *Knowledge Service* every minute in different attack scenarios. With more physical entities attacking the *MQTT Gateway*, more entities are affected, and more anomalies are reported in Z1. Up to 22 anomalies are detected every second. This number increases as we perform cross-zone attacks with more attackers (from Z2). The number of anomalies is proportional to the complexity of the network and the size of the affected area. Such overwhelming volume of anomalies is a challenge for security analysts in incorporating information from different sources (e.g., systems and security services) and manually processing anomalies to identify attack sources, impacts, and incident mitigation solutions. Developing an automated analysis can be complex when the network is changed in runtime (e.g., devices disconnect/re-connect and change IP addresses or when adding/removing devices). Mean-

[4]Note that in this study, our primary focus does not lie in evaluating the performance or capabilities of detection services. Rather, our emphasis is placed on enhancing explainability in security orchestration and analysis.

[5]https://support.vectra.ai/s/article/KB-VS-1233

while, in our framework, the *Incident Analysis* service enables individual anomaly processing as the tracing is performed based on the Environment Context. In the case of 3 cross-zone attackers, more than 2250 interactions with the *Knowledge Service* are performed every minute. Although there will be many duplicate traces because the Runtime Context remains unchanged for a certain period of time, they ensure that decisions are made in real-time with the most updated contexts (including physical system changes). The redundancy issue could be addressed using caches in our future work.

Listing 3: Excerpt of an incidentReport of a DDoS attack.

```
"reportID": "276852de-744a",
    "incidentInfo":{
        "incidentID": "T1498.002",
        "incidentName":"Network_Denial_of_Service",
        "runtimePattern": "HTTP Flood",
        "aggregatedMLMetric": {"minConfidence": 0.6, ...},
        ...},
    "incidentCause":{
        "47cf8b42-2eb4": {   # entityID
            "attribute": {
                "name": "mpo",
                "type": "factoryAsset"
                ...}}},
    "incidentConsequence":{
        "96er8s42-w2e6": {
            "attribute": {
                "name": "mqttGateway",
                "status": "disrupted",
                ...}},
        "0192eb5c-e33b": {
            "attribute": {
                "name": "S1",
                "type": "switch",
                "port": [2,3,5],
                "status": "overflow"}}
    },...
```

The incident report is presented entirely at the logical layer, as shown in Listing 3. We report the DDoS attack caused by the *MPO* using *HTTP Flood*, causing disruption in the MQTT gateway and overflow in switch S1. The *aggregatedMLMectric* allows security analysts to better assess the reliability and effectiveness of the ML models deployed in detecting security incidents. Without technical details (e.g., IP address and port), the report can be used to communicate between security teams with enhanced explainability. Security analysts can retrieve information related to the entities to validate the incidents and mitigation decisions After that, the *Security Plan Enforcement* service maps the logical entities (in digital twin) to their corresponding physical attributes in real-time (Section III-B3) before calling the APIs of the firewall and OPA engine. When an entity changes its IP address or port, these changes are promptly updated in the *Knowledge Service*. Firewall rules associated with the previous attributes are removed. New rules apply to the new attributes based on the corresponding entity's policies (from OPA). The entire process can be automated, thereby reducing human effort and errors.

TABLE III: Interaction with Knowledge Graph to obtain runtime context in different attack scenarios.

Number of Attackers	Runtime Context Update	Runtime Context Query	Cross-Zone Attack
1	886	630	No
2	1011	691	No
3	1170	760	No
1	1100	794	Yes
2	1220	846	Yes
3	1357	902	Yes

TABLE IV: Avg low-performance time of Frequent and Context-based (CB) re-training.

Re-training Method	Time Interval	Low-Performance State
Frequent Re-training	1 min	4.67 min
	2 min	5.52 min
	5 min	8.38 min
	10 min	15.85 min
RXOMS Re-training	Dynamic	4.21 min

C. Experiment 2: Incorporating domain-specific knowledge in ML-based security optimization

Although security services, such as ML-based services of Darktrace and Vectra AI, allow for re-training and fine-tuning ML models of detection services on user data, these optimizations are manually performed by security analysts based on their observation and feedback without considering runtime contexts. In this experiment, we aim to use Runtime Contexts combined with domain-specific knowledge to identify ML models that suffer from low performance, thus enabling us to re-train the models. We perform several attacks from the *Robot-VGR* to the two gateways with the number of network packets shown in Fig. 7. The amount of network traffic they generate is sufficient to disrupt the gateways. In the first 20 minutes, the entity only attacks one of the two gateways, after which it attacks both.

With a fixed number of network flows in SINDIT, each network flow is analyzed by an ML model trained with data from that flow. Without runtime context, we update all ML models periodically following certain time intervals, as shown in Table IV. With the unified *Knowledge Service*, the security analysts develop an automated analysis that knows exactly when the gateways are disrupted using the Environment Context, the network flows passing through the gateways using the Security Context, and the security services associated with such flows using the Tool-specific Context in real-time. Therefore, we are aware of which network flows will be affected by the incidents. If the ML models analyzing these flows fail to detect any anomalies, they are suffering from low performance. We estimate the average time these models spend in a low-performance state in Table IV. To address this issue, we re-train the model whenever low-performance states persist for more than 10 seconds. In both re-training methods, we re-train ML models using data from the nearest 5 minutes. With smaller time intervals, the average time spent in a low-performance state decreases significantly, but this incurs a heavier computational burden. Also, network flows in SINDIT

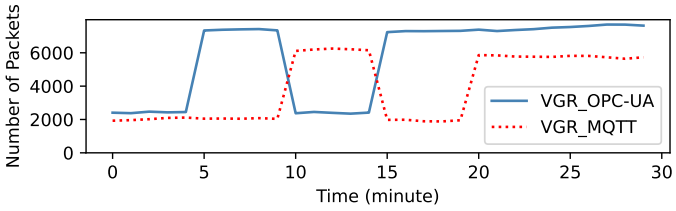

Fig. 7: Network packets from Robot-VGR to the two gateways.

evolve with various frequencies and patterns. Updating the ML models in all flows simultaneously more frequently does not achieve significant improvement where the time interval is smaller than 1 minute. Meanwhile, our re-training method relies on runtime context to maintain the performance of ML models for about 86% of the experiment time. The number of model updates is also less than frequent re-training (1-minute interval), approximately 4.2 times.

V. Related Work

A. Cybersecurity knowledge graph

As discussed in [16], several cybersecurity knowledge graphs have been developed to represent and visualize cybersecurity knowledge in a graphical format. For instance, the SEPSES knowledge graph [17] is a general-purpose cyber-knowledge graph that links and integrates vulnerabilities, weaknesses, and attack patterns from a wide range of data sources. CSKG is another example of a cybersecurity knowledge graph incorporating data from vulnerability and threat libraries. The Unified Cyber Ontology (UCO) is another graph modeling individual cyber security subdomains (e.g., cyber investigation, computer/network defense, threat intelligence, malware analysis, vulnerability research, and offensive/hack-back operations). However, these knowledge graphs still lack comprehensive coverage of concepts related to DT and ML-based security systems. To address this gap, our proposed graph model extends existing graphs with these relevant concepts. Our cybersecurity graph model is also enriched with tool-specific and DT knowledge. The developed knowledge graph is used to facilitate the cognitive processes of security specialists, enabling the utilization of runtime context in the diagnosis of cyber incidents and planning for respective mitigating actions. This integration ensures a holistic understanding of the cybersecurity landscape, enhancing explainability and contributing to a more robust and effective security framework.

B. Security orchestration and automation

The approach presented in [24] promotes security management and orchestration at an abstract level for software-defined infrastructure. It focuses on a conceptual design of a SOAR platform with its key dimensions but is generally applicable for the IT domain (services), not really for IoT or CPS domains. In [25], the authors propose a novel framework for automatic orchestration, configuration, and deployment of lightweight virtual network security functions in Unmanned Aerial Vehicles. The approach is applicable for the IoT domain, even though it mainly uses Software Defined Network (SDN) and Network Function Virtualization (NFV)-based security mechanisms, not any other security mechanisms. In [26], the authors present a semantic integration approach based on an ontological model to formalize the security tools, their capabilities, and their activities, enabling the automatic selection and integration of security tools. However, the ontological model solely focuses on security tools, lacking system context that could be provided by a DTs-based approach like ours. In [27], the proposed framework leverages DTs of IoT device modeling and their security features for security orchestration. For each IoT asset, there is one DT. Our work is built on a more systematic digital twin approach using knowledge graphs and DTDL. Moreover, we support explainability in automated security analysis and orchestration for DTs.

In general, these existing SOAR approaches do not leverage DT or tool-specific knowledge, thus, explainability is not comprehensively addressed. For complex systems, leveraging the DT approach can provide more complete orchestration and automation. Our approach allows security analysts to examine security operations/events within multiple security aspects and DT context, thereby enhancing explainability.

VI. Conclusion

In this paper, we have introduced a novel framework that offers enhanced explainability for security orchestration and analysis in DTs. Leveraging a centralized knowledge service, we support security unification and enrich security reports with insights derived from the DT and its security system. This facilitates automated security analysis and enforcement at the logical layer, thereby improving interoperability and communication between security teams and stakeholders. Additionally, our framework enables security analysts to integrate domain-specific knowledge in analyzing security operations/events and evaluating the performance of ML-based security services. Our framework has shown great potential in providing a holistic understanding across multiple aspects for security orchestration and analysis, especially within complex and dynamic DT environments. Future work can build on the centralized knowledge service as a comprehensive source of contextual data for developing ML-based security orchestration solutions.

VII. Acknowledgments

The research leading to this publication has partially received funding from the European Union's Horizon Europe research and innovation program under Grant Agreements 101070455 (DYNABIC) and 101058477 (COGNIMAN).

References

[1] C. Alcaraz and J. Lopez, "Digital twin: A comprehensive survey of security threats," *IEEE Communications Surveys & Tutorials*, vol. 24, no. 3, pp. 1475–1503, 2022.

[2] S. Suhail, M. Iqbal, R. Hussain, and R. Jurdak, "Enigma: An explainable digital twin security solution for cyber–physical systems," *Computers in Industry*, vol. 151, p. 103961, 2023.

[3] Q. Xu, S. Ali, and T. Yue, "Digital twin-based anomaly detection in cyber-physical systems," in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2021, pp. 205–216.

[4] C. Islam, M. A. Babar, and S. Nepal, "A multi-vocal review of security orchestration," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–45, 2019.

[5] Y. Wang, Z. Su, S. Guo, M. Dai, T. H. Luan, and Y. Liu, "A survey on digital twins: architecture, enabling technologies, security and privacy, and future prospects," *IEEE Internet of Things Journal*, 2023.

[6] W. Kritzinger, M. Karner, G. Traar, J. Henjes, and W. Sihn, "Digital twin in manufacturing: A categorical literature review and classification," *Ifac-PapersOnline*, vol. 51, no. 11, pp. 1016–1022, 2018.

[7] M. Farsi, A. Daneshkhah, A. Hosseinian-Far, H. Jahankhani *et al.*, *Digital twin technologies and smart cities*. Springer, 2020.

[8] T. R. Wanasinghe, L. Wroblewski, B. K. Petersen, R. G. Gosine, L. A. James, O. De Silva, G. K. Mann, and P. J. Warrian, "Digital twin for the oil and gas industry: Overview, research trends, opportunities, and challenges," *IEEE access*, pp. 104 175–104 197, 2020.

[9] A. Costantini, G. Di Modica, J. C. Ahouangonou, D. C. Duma, B. Martelli, M. Galletti, M. Antonacci, D. Nehls, P. Bellavista, C. Delamarre *et al.*, "Iotwins: Toward implementation of distributed digital twins in industry 4.0 settings," *Computers*, vol. 11, no. 5, p. 67, 2022.

[10] Z. Zhao, L. Shen, C. Yang, W. Wu, M. Zhang, and G. Q. Huang, "Iot and digital twin enabled smart tracking for safety management," *Computers & Operations Research*, vol. 128, p. 105183, 2021.

[11] P. Vähäkainu, M. Lehto, A. Kariluoto, and A. Ojalainen, "Artificial intelligence in protecting smart building's cloud service infrastructure from cyberattacks," *Cyber Defence in the Age of AI, Smart Societies and Augmented Humanity*, pp. 289–315, 2020.

[12] D. Vrandečić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Communications of the ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[13] T. Pellissier Tanon, G. Weikum, and F. Suchanek, "Yago 4: A reason-able knowledge base," in *The Semantic Web: 17th International Conference, ESWC 2020, Heraklion, Crete, Greece, May 31–June 4, 2020, Proceedings 17*. Springer, 2020, pp. 583–596.

[14] M. Waszak, A. N. Lam, V. Hoffmann, B. Elvesæter, M. F. Mogos, and D. Roman, "Let the asset decide: digital twins with knowledge graphs," in *2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*. IEEE, 2022, pp. 35–39.

[15] J. M. Gómez-Berbís and A. de Amescua-Seco, "Sedit: semantic digital twin based on industrial iot data management and knowledge graphs," in *Technologies and Innovation: 5th International Conference, CITI 2019, Guayaquil, Ecuador, December 2–5, 2019, Proceedings 5*. Springer, 2019, pp. 178–188.

[16] L. F. Sikos, "Cybersecurity knowledge graphs," *Knowledge and Information Systems*, pp. 1–21, 2023.

[17] E. Kiesling, A. Ekelhart, K. Kurniawan, and F. Ekaputra, "The sepses knowledge graph: An integrated resource for cybersecurity," in *International Semantic Web Conference*. Springer, 2019, pp. 198–214.

[18] Fischertechnik, "Training Factory Industry 4.0 24V with PLC connection board," 2023, accessed on 28-11-2023. [Online]. Available: https://www.fischertechnik.de/en/products/industry-and-universities/training-models/560841-training-factory-industry-4-0-24v-with-plc-connection-board

[19] N. Handigol, B. Heller, V. Jeyakumar, D. Mazières, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012.

[20] R. S. Framework, "Ryu SDN Framework component-based software defined networking framework build sdn agilely," 2023, accessed on 28-11-2023. [Online]. Available: https://ryu-sdn.org/

[21] OPA, "Open Policy Agent: Policy-based control for cloud native environments," 2023, accessed on 21-07-2023. [Online]. Available: https://www.openpolicyagent.org/

[22] J. Tang and H. Y. Ngan, "Traffic outlier detection by density-based bounded local outlier factors," *Information Technology in Industry*, vol. 4, no. 1, 2016.

[23] X. Tao, Y. Peng, F. Zhao, P. Zhao, and Y. Wang, "A parallel algorithm for network traffic anomaly detection based on isolation forest," *International Journal of Distributed Sensor Networks*, vol. 14, p. 1550147718814471, 2018.

[24] C. Islam, M. A. Babar, and S. Nepal, "Architecture-centric support for integrating security tools in a security orchestration platform," in *Software Architecture: 14th European Conference, ECSA 2020, L'Aquila, Italy, September 14–18, 2020, Proceedings 14*. Springer, 2020, pp. 165–181.

[25] A. Hermosilla, A. M. Zarca, J. B. Bernabe, J. Ortiz, and A. Skarmeta, "Security orchestration and enforcement in nfv/sdn-aware uav deployments," *IEEE access*, vol. 8, pp. 131 779–131 795, 2020.

[26] C. Islam, M. A. Babar, and S. Nepal, "Automated interpretation and integration of security tools using semantic knowledge," in *Advanced Information Systems Engineering: 31st International Conference, CAiSE 2019, Rome, Italy, June 3–7, 2019, Proceedings 31*. Springer, 2019, pp. 513–528.

[27] P. Empl, D. Schlette, D. Zupfer, and G. Pernul, "Soar4iot: securing iot assets with digital twins," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, 2022, pp. 1–10.