Journal of Information Security and Applications 89 (2025) 103956

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Severity-based triage of cybersecurity incidents using kill chain attack graphs

Lukáš Sadlek [a,*], Muhammad Mudassar Yamin [b], Pavel Čeleda [a], Basel Katt [b]

[a] Masaryk University, Brno, Czech Republic
[b] Norwegian University of Science and Technology, Gjøvik, Norway

ARTICLE INFO

Keywords:
Kill chain
Attack graph
Incident severity
Incident triage
MITRE ATT&CK
Cyber crisis

ABSTRACT

Security teams process a vast number of security events. Their security analysts spend considerable time triaging cybersecurity alerts. Many alerts reveal incidents that must be handled first and escalated to the more experienced staff to allow appropriate responses according to their severity. The current state requires an automated approach, considering contextual relationships among security events, especially detected attack tactics and techniques. In this paper, we propose a new graph-based approach for incident triage. First, it generates a kill chain attack graph from host and network data. Second, it creates sequences of detected alerts that could represent ongoing multi-step cyber attacks and matches them with the attack graph. Last, it assigns severity levels to the created sequences of alerts according to the most advanced kill chain phases that were used and the criticality of assets. We implemented the approach using the MulVAL attack graph generator and generation rules for MITRE ATT&CK techniques. The evaluation was accomplished in a testbed where multi-step attack scenarios were executed. Classification of sequences of alerts based on computed match scores obtained 0.95 area under the receiver operating characteristic curve in a feasible time. Moreover, a threshold exists for classifying 80% of positive sequences correctly and only a small percentage of negative sequences wrongly. Therefore, the approach selects malicious sequences of alerts and significantly improves incident triage.

1. Introduction

Security analysts must handle various types of cyber attacks, from adversarial scanning of network infrastructure to incidents that break the law. However, only a limited number of information technology (IT) and security specialists can address issues accompanying these attacks, and many alerts can be triggered by benign activities, which causes so-called alert fatigue [1,2]. Moreover, according to statistics, more than 40% of cloud security alerts are false positives or of low priority [3], and up to 30% of alerts are ignored [4]. Therefore, an essential task of security analysts is to triage these alerts based on the current security situation and assign them to roles qualified to resolve them based on their severity [5].

Cyber attacks are related to collocations of cyber threat scenarios, security events, and security incidents. The cyber threat scenarios are partially ordered sequences of threat events associated with some threat sources [6], representing scenarios of possible attacks. While threat events only describe possible adverse changes in security conditions, their actual changes are called security events and are often indicated by alerts from security tools. Security incidents adversely affect the network or its cyber assets [6]. In contrast, security events can lead to the effect, e.g., breach of confidentiality, integrity, and availability. Security incidents can cause a cyber crisis that jeopardizes the organization's objectives and viability and cannot be resolved by ordinary operations [7].

Creating cyber threat scenarios from security events can identify series of security incidents, i.e., multi-step attacks [8]. Related work (e.g., [1,9–11]) belongs to graph-based methods that use alerts as vertices and edges as their relationships, statistical methods coping with the distribution of historical data, machine learning focusing on features of alerts, and other methods [8]. They address challenges, such as hiding the attacker's actions, explainability, and knowledge base support [8]. Moreover, they cope with general issues of incident triage, namely lack of automation [12], lack of visibility [5], and dealing with too many alerts without having further context [13].

1.1. Research goals

The proposed triage approach processes security data that are outputs of security tools collecting data from networks and hosts and detecting cyber threats (see Fig. 1). These structured data with detection alerts would be otherwise processed by Security Operations Center (SOC) analysts directly [5]. The proposed approach discovers

* Corresponding author.
E-mail address: sadlek@mail.muni.cz (L. Sadlek).

https://doi.org/10.1016/j.jisa.2024.103956

Available online 16 January 2025

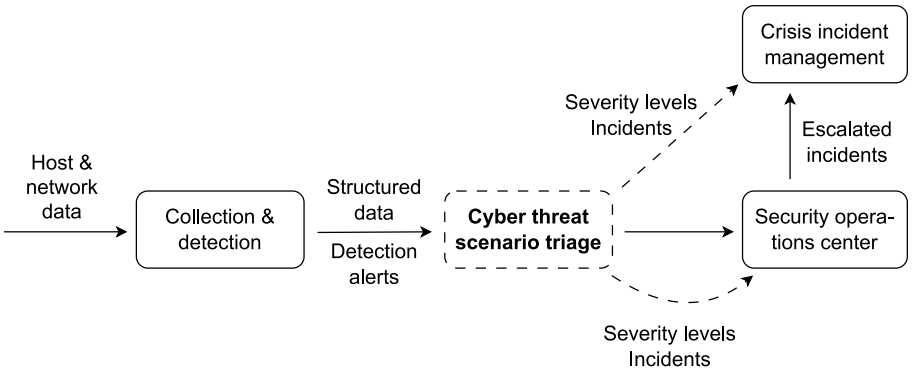


Fig. 1. Pipeline of processing security data until the security operations center or crisis incident management receives incidents. The proposed approach is represented by a dashed rectangle and its outputs by dashed edges.

severity levels and incidents (i.e., the most advanced attack techniques of ongoing cyber attacks) that can be further used by SOC or Crisis Incident Management (CIM), which could also receive them by manual escalation from SOC analysts. In this paper, we focus on two research questions:

1. *How can we triage possible sequences of detection alerts and consequently identify ongoing severe incidents based on host and network data?*
2. *Can triage of sequences of detection alerts based on host and network data lead to a realistic estimation of ongoing severe incidents (i.e., cyber crises)?*

1.2. Paper's contributions

We outline a new incident triage approach using custom kill chain attack graphs [14], MITRE ATT&CK [15], and the criticality of assets. The approach filters false positive random sequences of security alerts by matching to attack graphs that serve as templates of multi-step attacks, leading to future impacts on valuable targets. This approach provides several contributions. Other methods also provide them [1,9–11], but not all simultaneously.

The first contribution is that the attack graph also projects future actions, which is typical for methods that do not focus only on detected past alerts. The second contribution includes the ATT&CK knowledge base to classify the attacker's actions. The proposed approach triages alerts from eXtended Detection and Response (XDR). On the contrary, methods for event correlations often focus on Security Information and Event Management (SIEM) and Intrusion Detection System (IDS) alerts [8]. It aims to reveal cyber attacks with possible missing alerts. Another contribution is coupling the approach with cyber crisis management using proposed severity levels. The last contribution is the focus on lateral movement alerts that could significantly extend the sequences of alerts.

As a result, this paper can be relevant for *researchers in cybersecurity*, *cybersecurity managers*, *SOC analysts*, *incident handlers*, and members of *security teams*. The proposed approach can inspire them. They can extend their triage by kill chain attack graphs and adjust the approach in their working environment.

1.3. Paper's structure

This paper is organized in the following way. Section 2 provides background for security incident and cyber crisis management, cyber threat scenarios, and relevant data sources and tools. Section 3 overviews methods from the related work. Section 4 describes the design of the approach, e.g., how the severity levels were applied together with the MITRE ATT&CK tactics. Section 5 describes the proof-of-concept implementation that processes host and network data and detection alerts. Section 6 overviews the evaluation, network environment, results, and analysis of the approach's advantages and disadvantages. The final Section 7 concludes the paper.

2. Background

The background consists of four parts. First, we describe the background of security incident and cyber crisis management. The second part focuses on the triage of incidents. The third part concerns the identification of cyber threat scenarios necessary for the correlation of security events and the prediction of future progress of the security posture. The last part describes the necessary data sources and tools.

2.1. Security incident and cyber crisis management

Security incident management is usually provided by a Computer Security Incident Response Team (CSIRT) and a SOC. On the contrary, a crisis management team provides cyber crisis management. The CSIRT Services Framework [16] includes service areas called *information security event management* and *information security incident management* that provide analysis of security events and incidents, their triage, and response. Crisis management processes can run in parallel with incident handling. *Crisis management support* is a potential CSIRT service in a service area called information security incident management [16]. Its functions are mainly related to communication. However, the crisis management team should be able to use the resources provided by the CSIRT team to cope with the crisis [16]. This division of services indicates that the incident response often occurs in a single organization. Dealing with the crisis can also require more complex cooperation with other CSIRTs and organizations.

According to the European Union Agency for Cybersecurity (ENISA), incident handling is structured into detection, triage, analysis, and response, often followed by post-analysis and proposal of improvements [17]. On the contrary, the five tasks involved in crisis management are sense-making, meaning-making, decision-making, termination, and learning-and-reform tasks [7]. The sense-making phase controls whether the current situation crosses the border between ordinary detected incidents and the start of a crisis [7]. Both sets of tasks need to understand what caused the negative situation and resolve adverse security events with available resources.

Crisis management can be structured into several levels. Portesi and De Muynck [18] mention strategic, operational, and technical vertical levels. Østby et al. [19], Østby and Katt [20] divide responsibilities in the organization into strategic, tactical, and operational levels based on the Framework for Improving Critical Infrastructure Cybersecurity [21] applied to risk management. The framework defines the senior executive (i.e., strategic) level where mission priorities and overall risk management process are considered, the process (i.e., tactical) level that performs impact assessment, and the operational level

implementing the organization's operations to achieve desired mission requirements and outcomes. Therefore, Østby et al. [19] place organizations' SOCs at the operational level and ICT management teams that collaborate with security teams at the tactical level.

2.2. Incident triage

Incident triage belongs to essential tasks of the incident handling process. Analysts verify whether events are true positives, classify their severities based on their types (e.g., spam, scan, phishing), and assign responsibilities [22]. Triage is one of the daily tasks of SOC analysts automated in Security Orchestration, Automation, and Response (SOAR) technologies [23]. Moreover, it also determines whether the event is related to some of the previous events, its timeframe, category, and severity, and who should be responsible for its response [22].

Escalation of security events and incidents is essential for SOC. Ahmad et al. [5] describe that Level 1 (L1) analysts apply playbooks, guidelines, and experience to escalate the most severe incidents to L2 and L3 analysts and resolve the least severe incidents. Kokulu et al. [24] also confirm that incident triage is a task for human analysts. All analysts perceive and comprehend the security posture, while security leadership projects future security posture to provide the strategic management level [5].

SOC must deal with issues such as low visibility, automation, and speed of response [24]. The low visibility can be partially overcome by using various data sources, which implies a possible vast amount of data. Even though SOAR playbooks automate routine tasks, such as filtering and categorization of alerts, there is a need for a more sophisticated approach for triage that considers relationships between security events. According to Alahmadi et al. [13], fast validation of alerts must be reliable, explainable, analytical, contextual, and transferable because many of them were triggered by routine activities. Moreover, Advanced Persistent Threats (APTs) can replace at least one of their attack steps with new procedures that may not be detected. When using cyber threat intelligence, approaches should use Tactics, Techniques, and Procedures (TTPs) instead of low-level Indicators of Compromise (IoCs) to model threat actors' behavior [25,26].

An escalation ladder is a crisis-related approach to modeling different severity levels of ongoing situations [27]. According to it, preparing malicious activities (e.g., gathering credentials) and minor harassment before endangering peace in cyberspace can provide early warning signals about possible progress [27]. It was created by applying the Spectrum of Conflict [27], which expresses how crisis escalates during a conflict. However, other escalation models could also be easily adjusted to cyber operations. The Defense Condition (DEFCON) system has five levels — the lowest level of DEFCON 5 denotes peacetime, and the highest level of DEFCON 1 is related to the imminent danger of nuclear war [28].

2.3. Cyber threat scenarios

Kill chain and attack graphs are two threat models relevant to our paper. The kill chain model serves as a template for multi-step attacks. The popular *cyber kill chain* model consists of seven consequent phases — reconnaissance, weaponization, delivery, exploitation, installation, command-and-control, and actions on objectives [29]. Mechanisms and techniques usable for individual phases were discussed in [30]. The cyber kill chain was criticized because it is focused mainly on malware and provides no support for phishing attacks [31].

The *unified kill chain* [31], consisting of 18 phases, was proposed based on analyzing several kill chain types, including the cyber kill chain and MITRE ATT&CK. Currently, MITRE ATT&CK, released in 2018 [15], obtains extensive support from the cybersecurity community. When we interpret MITRE ATT&CK as a kill chain, we have techniques for individual phases called tactics (see also Section 2.4).

Attack graphs consist of vertices and edges with different meanings based on the type of the attack graph [32]. In our case, the vertices represent all information about the cyber threat scenario (e.g., privileges, properties of assets, and attack techniques), and edges denote the flow of actions. The attack techniques in the graph have incoming edges from their prerequisites and one outgoing edge representing their results. The attack graphs are applied, e.g., for security assessment [33], network hardening [34], and cybersecurity exercises [35].

Academic and commercial attack graph generators were compared by Yi et al. [36], e.g., open-source MulVAL [37] using logic programming and Cauldron [38]. These generators require suitable input to produce meaningful results [39]. The first group of inputs describes network topology and network security posture. The network security posture is expressed with vulnerability data, usually as Common Vulnerabilities and Exposures (CVE) entries [40]. The more problematic inputs are rules for generating attack graphs that define preconditions and results of attack steps since it is necessary to remove manually defined ad hoc rules for generating attack graphs [32].

2.4. Data sources and tools

Relevant data sources are MITRE ATT&CK and host and network data. MITRE ATT&CK [15] is a knowledge base providing TTPs. Columns of the ATT&CK matrix are tactics that can be interpreted as kill chain phases [31], e.g., privilege escalation and lateral movement. Rows describe techniques applicable within specific tactics, e.g., phishing belongs to the initial access tactic. ATT&CK techniques can be detected using several data sources allowing network security use, e.g., application logs and network traffic [41]. Execution of MITRE ATT&CK techniques can be accomplished by MITRE CALDERA [42].

Host and network data include, e.g., IP flows and system logs. IP flows contain packets with common properties, such as source and destination IP addresses, source and destination transport ports, and protocol, transmitted during a specific time window through a network observation point [43]. These data can be processed by IDS, which generates alerts when suspicious activity is detected.

The relevant tools for our paper are mainly CIM, SIEM, and Endpoint Detection and Response (EDR) tools. CIM tools belong to the strategic level of crisis management and facilitate crisis governance (e.g., tracking of incident's state), communication, and appropriate response. An example of a crisis management tool is Dispatch [44], which implements a workflow from creating security incident reports to their resolution and post-analysis.

SIEM tools are used for storing security data and may be extended with cyber threat detection. For example, Wazuh [45] is a security platform with prebuilt detection rules for the MITRE ATT&CK techniques. Data sources are typically integrated via lightweight data shippers called beats, e.g., Winlogbeat provides Windows system logs, and one of its modules processes logs from the System Monitor (Sysmon) service [46]. Unix-like systems can provide the data using the syslog format of log messages specified by RFC 5424 [47]. Network data containing IP flows, SSH accesses, and HTTP communication can be collected from IDS, such as a network analysis tool called Suricata [48].

The Wazuh platform [49] consists of four components — agents, server, indexer, and dashboard. Wazuh agents installed on managed endpoints send system and application data to the server. The Wazuh server analyzes these data and creates alerts when XDR functionality detects suspicious events. The Wazuh indexer stores these alerts in an index called *wazuh-alerts*. Detection can be extended with custom rules. All events are stored in the *wazuh-archives* index, including those not decoded. Last, the Wazuh dashboard monitors agents and visualizes security events.

3. Related work

The related work focuses on incident triage and cyber threat scenarios. Since this paper's approach is mainly graph-based, we focus on approaches that use them. The other types of methods are listed for completeness. A short comparison of our approach with the related work is included in Section 3.2.

3.1. Incident triage

Approaches for incident triage include, e.g., graph-based, attack model–based, Natural Language Processing (NLP), machine learning, and automata-based approaches. Graph-based approaches consider sequences of attack steps optionally representing longer time windows. Zhong et al. [50] used a centroid-based similarity measure applied on graphs where edges represented logical relationships and temporal continuity between constraints specified by analysts filtering relevant data. An example of such a constraint is an expression that the destination transport port equals 22.

Important subgroup contains approaches based on data provenance graphs that depict how and where pieces of data were created and modified using relationships as edges. Yuschan et al. [51] applied the provenance for threat discovery using dependency graphs containing processes, files, and sockets. The dependency graphs allow considering the necessary context for alerts, which was enriched by designed task deduplication, dividing the graph according to whether some vertices were already processed. The approach used system audit logs, also processed by detection systems that provided alerts. The result is a stream of alarms denoting the importance of alerts.

Similarly, Hassan et al. [2] used system event logs containing events related to processes, files, and sockets. Their control and data dependency paths contain processes as source entities and files, sockets, and processes as destination entities. As a result, they filter alerts based on anomaly scores above the defined threshold. Hassan et al. [9] also solved a similar problem as in this paper. They provided tactical provenance graphs that used the correct order of alerts according to the MITRE ATT&CK and ranked them. They revealed that 98% of these graphs are false positives.

Bryant and Saiedian [52] used the kill chain model to filter relevant SIEM alerts based on the kill chain phases and four domains — network, endpoint, domain, and egress. The model was evaluated by including it in SIEM software using pattern matching for fields inside of events and a series of conditions that had to be met by events and logs. Cyber Incident Handling Modelling Language by Mouratidis et al. [53] contains constructs allowing triage of incidents based on the criticality of assets (e.g., economic and service impact) and the severity of cyber threats based on threat models. The incident severity is based on impact (functional and informational), urgency, and recoverability.

Liu et al. [54] applied NLP in their Context2Vector approach, where representation learning on security events is followed by detecting deviation of the context, extraction of event features, and event ranking. Result labels and scores allow for consequent triage by SOC. Aminanto et al. [55] applied isolation forest to triage alerts from logs, and Fan et al. [56] used attention-based Long Short-Term Memory (LSTM) to predict future attack steps based on historical events. Ede et al. [57] used a deep learning approach to cope with numerous security events that had to be correlated with their context and triaged. The advantage of this approach is that it can reveal unknown evolving cyber threats using machine learning. However, using only detected security events is a disadvantage when security tools cannot detect some from the multi-step attack. It also focuses on events from one machine.

Wang et al. [1] proposed to use reinforcement learning for prioritization of alerts generated by network IDS and EDR tools. Used features, such as IP addresses, come from alerts. The context of alerts in their sequences comes from the cyber kill chain. The history of alerts is considered as well. Last, Zhong et al. [58] used automata for triage based on past analysts' actions (e.g., SELECT) and their temporal continuity. The state machine outputs instances of attack patterns found in intrusion detection system alerts, firewall logs, and vulnerability reports.

3.2. Cyber threat scenarios

The use of attack graphs is closely related to input data and classification of attack steps. For example, input data contain security alerts or indicators, and the classification can be ATT&CK. Milajerdi et al. [59] used a graph similarity metric based on a query graph containing relationships among IoCs from Cyber Threat Intelligence (CTI) reports and a provenance graph from kernel audit logs. Similarly, Milajerdi et al. [60] used system audit logs to create a provenance graph complemented by a scenario graph containing APT's TTPs from the kill chain. The final score was based on the likelihood of attack paths. The work by Nadeem et al. [11] is close to the triage of incidents since it constructs attack graphs based on alerts using suffix-based probabilistic deterministic finite automata. Their states belong to attack stages that correspond to different severity categories.

Hu et al. [10] applied attack graphs to construct attack scenarios using IDS alerts and the MulVAL attack graph generator. IDS alerts are mapped to the attack graph's vertices. Their work further extends attack graphs with data mining by clustering sequences of similar alerts. Data about attacker's behavior in honeypots were used by Mohammadzad et al. [61]. They proposed algorithms to prune unsuccessful attack paths and update the attack graphs with new changes. Tian et al. [62] constructed network attack paths from alerts. In their case, attack path creation was based on similarity metrics that they defined for properties of events, e.g., IP addresses. Alrehaili and Alshamrani [63] focused on attack graphs of advanced persistent threats. Their specific is that the method filters and clusters alerts that can probably be attributed to the same APTs before creating the attack graph.

Input for the approach by Mao et al. [64] contains IDS alerts from which IP flow attributes are extracted. A convolutional neural network filters false positive alerts, and an adjusted depth-first search is used to determine attack chains with phases from the kill chain model. Lyu et al. [65] also used IDS alerts and traffic data. Their abnormal states relationship graphs contain vertices representing flows and edges representing their relationships.

Approaches were evaluated in various settings and networks. Hassan et al. [9] processed Symantec EDR's alerts. They used 34 hosts that were actively used by the product development team. They also used three attack scenarios with 6, 13, and 20 attack techniques, where several attack techniques were repeated using different procedures. Holmes [60] used nine attack scenarios on seven hosts. Hu et al. [10] used seven hosts in one experiment and not a large network in the second one. Alrehaili and Alshamrani [63] used the DAPT 2020 dataset with public, private, log server, and gateway router virtual machines [66]. Mohammadzad et al. [61] used two hosts in the case study network.

Researchers already used ATT&CK in their threat models [67–69]. Wang et al. [70] used it to obtain attack paths of APTs based on alert and log correlation. The originality of the method consists of transforming alerts directly into graphs, applying community detection, and using Monte Carlo tree search. Another example used Common Attack Pattern Enumeration and Classification (CAPEC) [71], Common Weakness Enumeration (CWE) [72], and CVE [40] databases enriched with mitigations from MITRE ATT&CK solely to provide attack graph generation and optimization of possibly employed controls according to cost [73].

Commercial EDR tools can also create cyber threat scenarios containing ATT&CK techniques. For example, Kaspersky EDR can analyze attacks and create activity trees with ATT&CK techniques [74]. Symantec EDR visualizes multi-step attacks using MITRE ATT&CK [75]. Moreover, EDRs (e.g., in Cisco Secure Endpoint [76]) can also provide

containment and investigation functionality, focusing on past actions from the data.

The difference between the proposed approach and the related work is as follows. The approach provides all of the contributions listed in Section 1.2 simultaneously. In contrast, related research works provide subsets of them. The approach advances research works from Sections 3.1 and 3.2 due to the inclusion of MITRE ATT&CK techniques and MITRE D3FEND's Digital Artifact Ontology in kill chain attack graphs introduced in Section 4.1 [14,77]. The inclusion implies other future advantages mentioned in Section 4.1, such as using large language models that already contain these knowledge bases in their corpus. Another advancement is the suitability of the proposed method for triaging XDR alerts. The related work often focuses on IDS alerts [10,64,65]. The XDR provides additional functionality to IDS, e.g., determining ATT&CK techniques, while the related work for IDS does not mention any relationship to the ATT&CK knowledge base. Estimating the future steps of attackers with the help of attack graphs is the last advancement. Creating sequences of detected XDR alerts alone does not provide information about whether the last alerts from these sequences could result in the follow-up steps, including dangerous impacts.

4. Design of the approach

In this section, we describe the design of the approach that creates cyber threat scenarios using kill chain attack graphs from host and network data. It matches sequences of detected alerts called *evidence paths* to these scenarios to identify possible ongoing multi-step cyber attacks. As a result, it determines severity levels for incident triage and the most advanced incidents from the evidence paths.

The first part of this section explains attack graphs and types of their vertices. The second part describes the design of severity levels that aims to provide compatibility with the cyber escalation ladder [27] used for crisis management. The last part explains the design of the Cyber Threat Scenario (CTS) Analyzer [78], which is a proof-of-concept implementation of the proposed approach.

4.1. Kill chain attack graphs

A *kill chain attack graph (KCAG)* is a graph with five types of vertices: attacker's capabilities of asset control (A) (called levels of asset control in [14]), properties of assets (P), countermeasures (C), attack techniques (T), and attack goals (G) [14]. More formally, KCAG is a pair of vertices V and edges E denoted as (V, E) where $V = A \cup P \cup C \cup T \cup G$ and $E = E_p \cup E_r$. Two types of edges represent prerequisites of attack techniques defined as $E_p = \{(u, v) \mid u \in A \cup P \cup C \wedge v \in T\}$ and results of attack techniques defined as $E_r = \{(u, v) \mid u \in T \wedge v \in A\}$. Moreover, an attack path is defined as an alternating sequence of the attacker's capabilities and techniques until an attack goal is reached, i.e., $(a_1, t_1, a_2, t_2, \ldots, a_n, t_n, g)$ where $\forall i \in \{1, \ldots, n\} : a_i \in A \wedge t_i \in T$ and $g \in G$. Techniques from the definition are also called attack steps.

The capabilities of asset control are always related to specific cyber assets, e.g., a network service or a file. They include security properties breached to obtain control over the assets by techniques. These security properties are obtained from the STRIDE model — authentication, integrity, non-repudiation, confidentiality, availability, and authorization [79].

Properties of assets describe properties, such as unpatched vulnerabilities, open ports, and allowed access between hosts. Countermeasures are always related to attack techniques since their absence allows attack techniques to be executed. They always appear in KCAGs as negations. Attack techniques were obtained from the MITRE ATT&CK knowledge base. Each attack technique can be classified according to the STRIDE model into six threat categories — spoofing, tampering, repudiation, information disclosure, denial of service, and elevation of privileges [79].

Each technique has its prerequisites and results specified by generation rules. One rule for an attack technique t is defined by a list of prerequisites that can form with t an edge from E_p and a result forming with t an edge from E_r. Prerequisites for attack techniques always include one attacker's capability of asset control and optional asset properties and not employed countermeasures. KCAG contains only one result for each attack technique. The result is always a capability of asset control with security property breached by the attack technique, e.g., breached availability for a denial of service. An example of a rule for the technique *Exploit Public-Facing Application* (T1190) is shown in Fig. 2. It shows examples of the predicates *application*, *vulnerableAsset*, *networkService*, and *networkConnection*.

The rule in Fig. 2 expresses that the attack technique will violate the confidentiality of *Software* on host *H* (see *application*) when there is vulnerable software *Software* that can be remotely exploited to cause confidentiality loss (see *vulnerableAsset*). Moreover, the software is accessible as a service on port *Port* using *Protocol* (see *networkService*), and the attacker has already violated authentication of the network connection to the port and the protocol (see *networkConnection*). Since MulVAL uses logic programming, the rule enforces the same value of variable *Software* in all predicates. Moreover, an underscore in *vulnerableAsset* denotes that there can be any value. A string starting with a lowercase letter is a constant, e.g., *remote*.

The *application* and *networkConnection* in Fig. 2 are the attacker's capabilities of asset control. They contain number 2, denoting that security properties based on the STRIDE model [79] – confidentiality and authentication – were violated. Number 1 would mean that the attacker revealed the asset's existence, and number 0 would mean that the attacker does not know the asset exists [14]. Predicates *vulnerableAsset* and *networkService* are properties of assets.

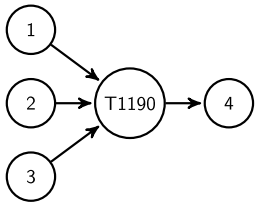
Attack goals are, in fact, the attacker's capabilities of asset control but always appear at the end of attack paths. Chaining of individual attack steps in the attack graph uses asset types and their security properties violated by attack techniques that are, at the same time, prerequisites of the next techniques on attack paths. A KCAG generated during the implementation of the approach is depicted in Fig. 5.

KCAGs aim to advance the existing attack graph models concerning the automation of attack graph generation since the ATT&CK knowledge base, D3FEND knowledge graph [80], and CVE are prominent sources for KCAGs. For example, countermeasure vertices can be obtained from the D3FEND knowledge graph and ATT&CK's mitigations, while categories of cyber assets can be obtained from D3FEND's Digital Artifact Ontology [81]. Using large language models (LLMs) could allow for the creation of rules for attack techniques. The first works about using LLMs for threat models have already appeared, e.g., [82]. Moreover, using ATT&CK and D3FEND as sources may make KCAGs more community-friendly, and KCAGs can be shared using Attack Flow language [83]. ATT&CK also allows coupling with CTI and EDR alerts [74,75]. More information about KCAGs can be found in our previous work [14].

4.2. Severity levels of cyber threat scenarios

Severity levels conform to the cyber escalation ladder [27] relevant for crisis escalation and are inspired by the DEFCON states [28]. It allows not only SOC but also crisis management to use the results of incident triage. Fig. 3 visualizes proposed severity levels divided according to the criticality of controlled assets and the category of possible actions on them. Moreover, Table 1 assigns the actions (i.e., tactics) from the MITRE ATT&CK version 12 [15] to the severity levels according to whether they represent initial access, misuse, or impact and exfiltration related to assets (see Fig. 3). These levels are applied to match evidence paths to attack graphs in Section 5.

The criticality of assets is expressed using the onion representation of the defense-in-depth model [84]. Noncritical assets are often accessible from the Internet, depending on their use cases, and critical assets



ID	Predicate
1	vulnerableAsset(*H, Software, _, remote, appConfidentialityLoss*)
2	networkService(*H, Software, Protocol, Port, _*)
3	networkConnection(*2, authentication, H, Protocol, Port*)
4	application(*2, confidentiality, H, Software*)

Fig. 2. An attack graph excerpt created when a rule for attack technique Exploit Public-Facing Application (T1190) is fulfilled. Prerequisites and one result are listed in the table.

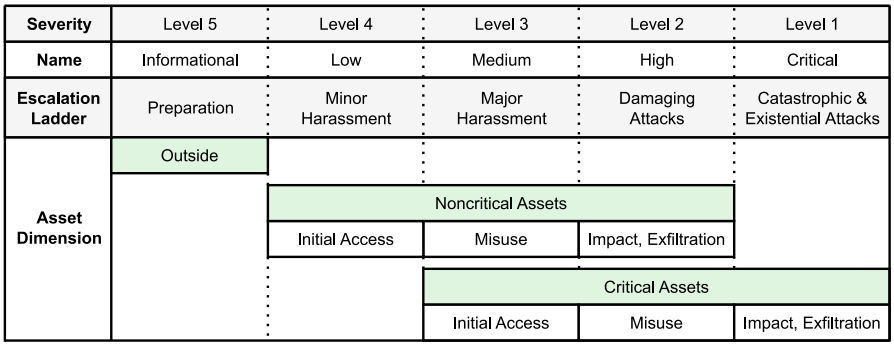


Fig. 3. Comparison of the cyber escalation ladder, the criticality of assets, and severity levels. Level 5 represents the lowest severity, and Level 1 the highest severity.

Table 1
ATT&CK tactics are divided into severity levels according to asset criticality and the category of actions depicted in Fig. 3 in the asset dimension.

Severity level	Description	ATT&CK tactic	Asset
5	Usual operations	Reconnaissance Resource Development	any
4	The least severe incidents	Initial Access	noncritical
3	Immediate danger of a severe incident	Initial Access	critical
		Collection Command and Control Credential Access Execution Privilege Escalation Defense Evasion Persistence	noncritical
		Lateral Movement Discovery	any
2	More severe incidents	Collection Command and Control Credential Access Execution Privilege Escalation Defense Evasion Persistence	critical
		Impact Exfiltration	noncritical
1	The most severe incidents	Impact Exfiltration	critical

only from the internal network. The initial access category in Fig. 3 should contain attack techniques that allow for establishing an initial foothold. The impact category should contain techniques that breach the confidentiality, integrity, and availability (CIA triad) of an attack goal. The misuse category should contain techniques used after the initial access and before the final impact.

The lowest severity level (Level 5) corresponds to the *preparation* phase of the cyber escalation ladder without any adversarial control of the organization's assets. It contains reconnaissance and resource development during usual operations, as shown in Table 1. Level 4 represents incidents related to the *minor harassment* category. It contains the initial access tactic from ATT&CK. Since [27] mentions that gathering credentials belongs to minor harassment and hence to Level 4, we must clarify that such gathering of information belongs to the reconnaissance in ATT&CK [15]. The credential access tactic contains more sophisticated techniques for stealing credentials. Most of them require previous access to the network and sufficient privileges obtained by the attacker.

Level 3 describes less critical incidents related to the *major harassment* category that openly manifest malicious behavior. The attacker vertically extends the control by breaching different security requirements. The attacker also uses noncritical assets to access other assets by lateral movement, simultaneously representing initial access to them (see Table 1). Therefore, we merged misuse of noncritical assets with initial access actions to critical assets in Fig. 3 because the latter is often closely coupled with the former. Discovery is also classified into this level for any asset because it precedes lateral movement and reveals additional information about the system and the network. The difference between discovery and reconnaissance is that the discovery is accomplished within the network.

At Level 2, the attacker can accomplish minor and major *damaging attacks*. Fig. 3 contains, at this level, the impact on noncritical assets and techniques for misusing critical assets. Kostyuk et al. [27] list the destruction of noncritical data and targeted harassment of infrastructure as examples of cyber actions belonging to damaging attacks. There is a high probability that they are targeted because the attacker must have enough skills to break through the defense in the organization. Attack statistics confirm it, e.g., scanning is detected more often than other kill chain phases [11].

Level 1 denotes that the most severe incident impacting the organization's critical assets has happened. It corresponds to *catastrophic and existential attacks* based on the cyber escalation ladder [27] because it will cause the organization's mission not to be fulfilled on the tactical level of risk management. The impact tactic in Table 1 allows the attacker to breach the integrity and availability of cyber assets, while the exfiltration tactic allows for impacting confidentiality.

When considering MITRE ATT&CK tactics, we focus mainly on the host level of assets since categorizing phases of a kill chain model into

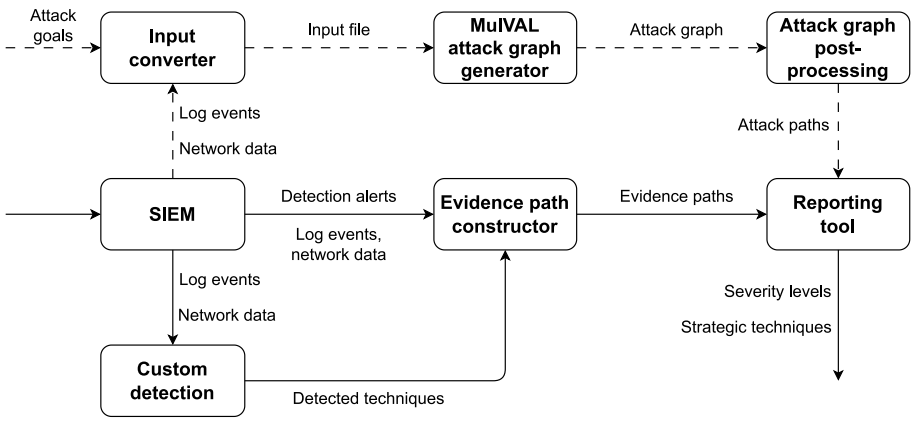


Fig. 4. Design of the prototype implementation of the CTS Analyzer. Arrows depict the processing of data by components. Dashed lines describe attack path processing functionality, while solid lines represent evidence path processing functionality and shared parts.

severity levels does not provide enough granularity for other assets (e.g., files and processes). If needed to work with other categories of assets, techniques must be categorized individually since each can target a different type of asset. However, the more granular types of assets can still be considered in kill chain attack graphs (see Section 5). Introducing these details into triage would cause it to be too complicated, not transparent, and not quickly understandable by a security expert.

4.3. Architecture of the cyber threat scenario analyzer

The proposed approach is divided into two stages. The first creates a kill chain attack graph representing a possible cyber threat scenario. The second creates evidence paths (i.e., sequences of detected alerts). Finally, results from these two stages are combined by path matching that determines whether the evidence path resembles an attack path. The detailed workflow of the approach is depicted in Fig. 4.

The kill chain attack graph is created according to log events and network communication using the MulVAL attack graph generator. It consists of three components — an *Input converter*, a *MulVAL attack graph generator*, and *Attack graph post-processing*. These three components were inspired by the design of the proof-of-concept implementation [85] described in the paper about kill chain attack graphs [14].

The next stage creates evidence paths consisting of attack techniques detected by the SIEM and the component *Custom detection*. The evidence paths are ordered according to the timestamps by the component *Evidence path constructor*. The combination of results from these two stages is done using the *Reporting tool*. The components called *Custom detection*, *Evidence path constructor*, and *Reporting tool* provide new functionality. Components from Fig. 4 are represented by Python modules in the implementation (see Section 5).

The approach begins in the *SIEM*, which denotes data collection and threat detection functionality for simplicity. It provides necessary data from log events, network traffic, and Network Intrusion Detection System (NIDS) messages. These data contain principal system events, communication of IP addresses, and detection alerts. A necessary assumption for the methodology is that these alerts must contain ATT&CK IDs, tactics, timestamps, and IP addresses of originating agents, as can be obtained from the state-of-the-art EDR tools from Section 2. The *Input converter* converts the information from the SIEM into an input that the MulVAL attack graph generator can process. The input file should contain facts expressed according to predicates and rules.

As a next step, the *MulVAL attack graph generator* processes the input file, ruleset defining rules for attack techniques that can appear in the attack graph, and information about attack goals. Its reasoning algorithm generates possible attack graphs for the attack goals. The created attack graph contains attack paths organized according to the kill chain because of the ruleset prepared according to the KCAG methodology. However, post-processing is needed to remove unnecessary kill chain phases for attack techniques and to assign types of KCAG's vertices by the *Attack graph post-processing* component. Its intermediate results are individual attack paths.

Custom detection should apply additional detection rules to raw data to reveal attack techniques not detected by the SIEM's ruleset. For example, we can use Windows event IDs to detect deleting and renaming files. Other examples include the termination of services and the installation of software. The detection alerts are then passed to the evidence path constructor.

The *Evidence path constructor* deals with the large amount of data that can be obtained from SIEM and determines evidence paths. These evidence paths allow the filtering of false positive alerts by providing context using the kill chain model and its phases for detected attack techniques. However, it is necessary to generate the evidence paths optimally and not have an exponential count of possible evidence paths.

The *Reporting tool* accomplishes path matching of evidence paths to attack paths to reveal whether necessary steps from evidence paths are present in the attack paths. The evidence path may not be meaningful compared to the generated attack path. We could obtain evidence paths with kill chain phases that conform to the ordering, but their chaining makes no sense in reality, e.g., prerequisites and postrequisites of individual techniques were not met. In this case, the MulVAL generator will not generate the attack path. Finally, the severity levels are determined for IP addresses based on their evidence paths, which consist of attack techniques. The tool can also output additional information, e.g., strategic techniques in these attack paths and countermeasures that destroy the maximum count of attack paths. In such a case, the techniques must be present in all attack paths to a single goal, and countermeasures must be present in all attack paths in a graph.

5. System prototype

The proposed approach and components from the designed architecture of the CTS Analyzer were implemented in Python (see supplementary materials [78]). For this purpose, we also used CALDERA and Wazuh. The implemented approach constructs attack paths and evidence paths. Attack paths lead from leaves to the goal in the attack graph created by MulVAL. Evidence paths are created based on detected alerts. Severity levels are determined from evidence paths that resemble attack paths, i.e., multi-step attacks, for the most advanced and detected techniques applied by attackers.

Algorithm 1: Identification of Attack Paths

```
Input  : log_file, ruleset, attack_goals, end_timestamp
Output: techniques_from_paths
/* A part of the algorithm that belongs to the input
   converter component in Figure 4.                    */
1  input_file = [ ];
2  for event ∈ log_file do
3  |  if event.timestamp < end_timestamp then
4  |  |  lines = determine_facts(event);
5  |  |  input_file.add(lines);
6  |  end
7  end
8  lines = determine_facts(attack_goals);
9  input_file.add(lines);
/* The pseudocode of generating attack graphs by MulVAL
   is reported in Figure 6 in [86].                    */
10 attack_graph ← run_mulval(input_file, ruleset);
/* Attack graph post-processing from Figure 4. The
   result contains attack paths represented by
   extracted techniques.                               */
11 for vertex ∈ attack_graph do
12 |  assign_type(vertex);
13 end
14 techniques_from_paths ← [ ];
15 for goal ∈ attack_graph do
16 |  for path ∈ simple_paths(attack_graph, attack_graph.start, attack_goal) do
17 |  |  determine_kill_chain_phases(attack_graph, path);
18 |  |  techniques_from_path ← [ ];
19 |  |  for vertex ∈ path do
20 |  |  |  if vertex.label == "TECHNIQUE" then
21 |  |  |  |  techniques_from_path.add(vertex);
22 |  |  |  end
23 |  |  end
24 |  |  if techniques_from_path ∉ techniques_from_paths then
25 |  |  |  techniques_from_paths.add(techniques_from_path);
26 |  |  end
27 |  end
28 end
29 return techniques_from_paths;
```

5.1. Identification of attack paths

The identification of attack paths was implemented using Wazuh SIEM [45], which can be configured to collect data in syslog format and Windows log events from agents. Moreover, Suricata [48] integration in Wazuh allows for monitoring and analyzing network traffic processed by a network interface of an endpoint. In our case, we used network communication obtained from Suricata messages captured on a specific host and system logs from *wazuh-alerts* and *wazuh-archives* indices to create an input file for the MulVAL generator. It is schematically depicted in the first for loop of Algorithm 1 on Lines 1–7.

The ruleset file contains rules and related predicates expressed using the KCAG methodology [14]. It contains one or multiple rules for each ATT&CK technique. Our ruleset was partially inspired by the default MulVAL ruleset from [37], but we retained only such predicates that conform to the methodology. In other cases, we created rules for ATT&CK techniques we wanted to consider. The predicates allow information from Wazuh SIEM to be expressed as facts.

System logs allow the operating system to be determined according to *"windows"* or *"syslog"* tags indicating Windows and Linux operating systems. In such a case, the input file contains the fact *installed(ip_address, windows)* to express that the Windows operating system is installed on the host with the IP address. Moreover, accounts are sometimes listed in the data, e.g., SYSTEM for Windows. Therefore, we can add the *hasAccount* facts denoting either the *"root"* or the *"user"* capabilities of asset control on the host into the input file. Allowed network access between two IP addresses and their ports is determined according to past network communication. Similarly, a network service existing on a host and port is determined from SIEM data. We add all IP addresses to their */24* subnet, but other subnets are also possible.

Each rule in the ruleset has prerequisites and a result. One type of the possible prerequisites corresponds to negations of facts that *"countermeasure X was applied"*. In other words, if a countermeasure is applied on a host based on the input file, the technique cannot appear in KCAG. If not, the technique can appear. The ruleset and countermeasures for techniques in the ruleset were specified manually.

We applied one adjustment to decrease the size of attack graphs. The input file does not contain port numbers because their source could be only Suricata warnings and alerts that complained about possible issues. However, we could still generate the graph without using port numbers as a template for mapping detected alerts. Nevertheless, when a rule for an attack technique required a specific port number, the generation algorithm propagated it through the attack graph and enforced its use. For example, port number 22 was used for SSH service. Other details on how the input files are created can be found in *input_convertor.py* in the supplementary materials [78].

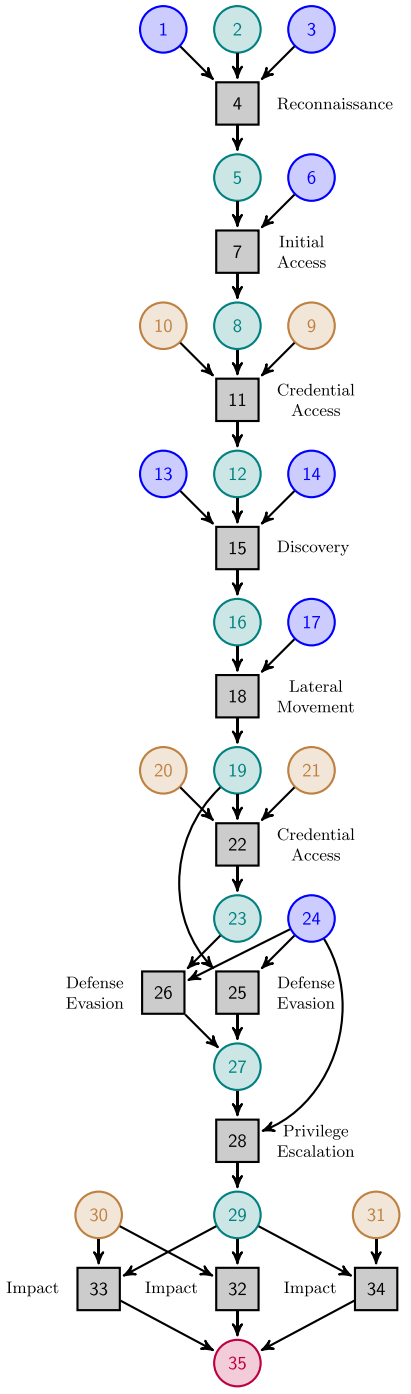
The last necessary input for MulVAL contains attack goals (see Lines 8 and 9). In the system prototype, the goals are the IP addresses of network hosts. In addition, one predicate must specify the attack source. For example, during the development, we used the IP address of the local CALDERA command-and-control server that emulates adversary behavior. This information is expressed by the *externalActor* predicate in the example attack graph in Fig. 5, which models a scenario where the attack could originate from the CALDERA machine with IP address *10.255.11.28*. In practice, the attacker's location would be the Internet.

The KCAG is generated in two steps. First, MulVAL generates attack graphs based on the ruleset containing attack techniques but no tactics (i.e., kill chain phases) and no types of vertices on Line 10 [86]. The component of attack graph post-processing adds types of vertices on Line 12 to obtain representation in Fig. 5. The component uses a dictionary in the proof-of-concept implementation, which contains tactics for each ATT&CK technique obtained from the ATT&CK knowledge base. Each attack technique is assigned one or multiple tactics from ATT&CK. For example, T1078 (*Valid Accounts*) belongs to Defense Evasion, Persistence, Privilege Escalation, and Initial Access [15].

Consequently, the ordering of ATT&CK tactics from Fig. 6 is applied as a dictionary of successors in the proof-of-concept implementation. It represents the ordering of kill chain phases in cyber threat scenarios used by attack graph post-processing. A unidirectional edge represents that one phase follows after the other. For example, the lateral movement phase will follow after the discovery phase. However, an attack can entirely skip the discovery phase. In such a case, the lateral movement must transitively fulfill the constraints imposed on the Discovery vertex and follow after, e.g., initial access. Bidirectional edge represents that tactics can happen in any order, e.g., reconnaissance and resource development. However, they both must precede initial access.

All tactics are related to one host except for reconnaissance and resource development, which are related to the whole network. Returning to reconnaissance or resource development would start another attack path. Exfiltration, impact, and lateral movement end paths on individual hosts. The ordering of ATT&CK tactics from Fig. 6 is used to check the correctness of phases assigned to two subsequent attack techniques in a cyber threat scenario from MulVAL. All tactics that breach the ordering of tactics are automatically removed on Line 17. The last steps of Algorithm 1 extract attack techniques from attack paths to provide input for matching evidence and attack paths.

Fig. 5 contains a KCAG path from network scanning to the final impact phase, including lateral movement. Ordering of techniques has the following meaning. The attacker scans network services first. Using valid accounts and brute force, the attacker can obtain user



ID	Value
1	networkService(*10_11_2_113*, _, *tcp*, _, _).
2	externalActor(*10_255_11_28*).
3	accessAllowed(*10_255_11_28*, *10_11_2_113*, *tcp*, _).
4	T1595 – Active Scanning
5	networkConnection(*2*, *authentication*, *10_11_2_113*, *tcp*, _).
6	hasAccount(_, *user*, *10_11_2_113*, *windows*).
7	T1078 - Valid Accounts
8	account(*2*, *authentication*, *user*, _, *10_11_2_113*, *windows*).
9	multifactorAuthentication(*no*).
10	strongPasswordPolicy(*no*).
11	T1110 – Brute Force
12	account(*2*, *authentication*, *root*, _, *10_11_2_113*, *windows*).
13	networkService(*10_11_1_10*, _, *tcp*, _, _).
14	accessAllowed(*10_11_2_113*, *10_11_1_10*, *tcp*, _).
15	T1046 – Network Service Discovery
16	networkConnection(*2*, *authentication*, *10_11_1_10*, *tcp*, _).
17	hasAccount(_, *user*, *10_11_1_10*, *windows*).
18	T1021 – Remote Services
19	account(*2*, *authentication*, *user*, _, *10_11_1_10*, *windows*).
20	multifactorAuthentication(*no*).
21	strongPasswordPolicy(*no*).
22	T1110 – Brute Force
23	account(*2*, *authentication*, *root*, _, *10_11_1_10*, *windows*).
24	installed(*10_11_1_10*, *windows*).
25	T1112 - Modify Registry
26	T1112 - Modify Registry
27	file(*2*, *integrity*, *10_11_1_10*, *registry*).
28	T1543.003 – Windows Service
29	system(*2*, *authorization*, *10_11_1_10*, *root*).
30	dataBackup(*10_11_1_10*, *no*).
31	encryptedDisk(*10_11_1_10*, *no*).
32	T1486 – Data Encrypted for Impact
33	T1485 – Data Destruction
34	T1565.001 – Data Manipulation – Stored Data Manipulation
35	file(*2*, *integrity*, *10_11_1_10*, _).

Fig. 5. Attack graph containing attacker's capabilities with green color, techniques as black squares, countermeasures with brown color, properties of assets with blue, and the attack goal with purple. Duplicated vertices (with IDs 9, 10, 20, and 21) simplify visualization.

privileges and escalate to root privileges on the host with IP address *10.11.2.113* accessible from the CALDERA command-and-control server. Consequently, the discovery of network services and misuse of existing remote services would allow the attacker to move laterally and escalate the capabilities of asset control on a new host with IP address *10.11.1.10*. Afterward, the attacker could use Windows-specific techniques (T1112, T1543.003) and conclude with three options for the final impact.

Fig. 5 contains countermeasures not employed for a specific host or network environment. Asset properties describe network services, allowed access between IP addresses, existing accounts on hosts, and installed software. The attacker's capabilities of asset control express that the attacker established a network connection, obtained user and root privileges, breached the integrity of a registry, and created or modified Windows services without being authorized to it prior to the attack. In this demonstrative example, the final goal is the breached integrity of a file on the second host.

5.2. Identification of evidence paths

The construction of evidence paths processes detection alerts with ATT&CK IDs, names of techniques, tactics, timestamps, and IP addresses of originating agents. We enrich these events with indirect detection alerts by providing detection rules for those attack techniques

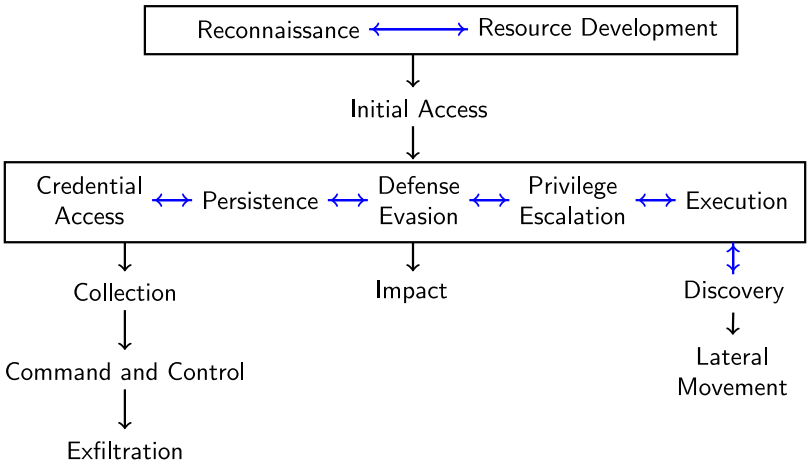


Fig. 6. Ordering of ATT&CK tactics for cyber threat scenarios with one host. Lateral movement will lead to another host, while exfiltration and impact end the scenarios. Each arrow represents that the end tactic can follow after the start tactic.

that SIEM's ruleset cannot detect. These indirectly detected events have the same output format as alerts provided by the SIEM.

In the beginning, we determine which techniques can be starting techniques of the evidence paths. In our case, we allow the starting techniques from one of the tactics: reconnaissance, resource development, initial access, and lateral movement, which is also initial access to internal hosts. Consequently, we process all events obtained from SIEM, ordered according to their timestamps, and divided for individual IP addresses in a dictionary. We apply aggregation functionality to reduce the large number of alerts by passing through the list of events for individual IP addresses. During one pass, we replace sequences containing several occurrences of the same event with one occurrence and the count of how many times it appeared.

Particular evidence paths on one IP address are constructed in the following way. We aim to create the longest allowable evidence paths that conform to the ordering of phases in Fig. 6. If there is no evidence path, we will create it. If there are already some evidence paths, then we check all existing paths to determine whether the current alert can be added at its end according to its kill chain phases. If not, we will iterate backward to determine the correct position where the technique could be placed. In such a case, we create a new evidence path containing the first part of the evidence path and the new technique. Otherwise, we must create a new evidence path if this technique can start it, according to the technique's tactic. It is necessary to check for duplicates rather than creating already existing evidence paths. The identification of evidence paths is implemented in *evidence_path.py* [78].

5.3. Matching of paths

SIEM data can contain false positive alerts, e.g., reconnaissance or regular deletion of files on critical systems during system maintenance by an administrator. Therefore, it is necessary to have some estimated cyber threat scenario (i.e., attack path) and compare it with the evidence path constructed from SIEM alerts using path matching. Moreover, sophisticated APTs may accomplish attack steps that span long time windows or are not detected. Hence, it is necessary to cope with situations when some alerts are missing from the scenario, and the comparison should not be exact.

The matching of the evidence paths to the attack paths works in the following way (see Fig. 7 and Algorithm 2). Attack paths, evidence paths, and a threshold are necessary inputs. The path matching iterates through the evidence paths and tests each technique to determine whether it can be mapped to a technique from the attack path (see Line 5 in Algorithm 2 and the whole of Algorithm 3). The attack path may contain more techniques, but a subset of techniques from the evidence path must be present within the attack path in the correct order. Therefore, the approach will filter a lot of simple false positives because it provides context and meaning for the threat alerts. The match score is computed as a maximum ratio of the count of alerts found in some attack path to the length of the evidence path (i.e., count of all alerts) using the following Eq. (1):

$$score(alerts, graph) = \max_{path \in graph} \left(\frac{count\_of\_found(alerts, path)}{count(alerts)} \right) \quad (1)$$

The match score has values from 0.0 to 1.0. Its application can be explained using Fig. 7. The evidence path contains three alerts, all found in the attack path, resulting in a match score of 100%.

A threshold imposed on the match score can make filtering evidence paths more strict. When the threshold is zero, we accept all evidence paths with the correct ordering of kill chain phases.

A particular case is the lateral movement tactic. Evidence paths contain detection alerts only from individual hosts, and lateral movements can be starting tactics. During the path matching, attack paths are divided into parts belonging to individual hosts by lateral movements

Algorithm 2: Matching of Evidence and Attack Paths

Input : evidence_paths, attack_paths, threshold = 0

`/* events and severity levels */`

Output: results

```
1  results ← {};
2  for ip_address ∈ evidence_paths do
3      for evidence_path ∈ evidence_paths[ip_address] do
4          for attack_path ∈ attack_paths do
5              for event ∈ check_sequence_in_path(evidence_path, attack_path,
                 ip_address, threshold) do
6                  if severity_level(event) < severity_level(results[ip_address])
                     then
7                      results[ip_address] ← (event, severity_level(event));
8                  else if severity_level(event) == severity_level(results[
                     ip_address]) then
9                      results[ip_address].add((event, severity_level(event)));
10             end
11         end
12     end
13 end
14 return results;
```

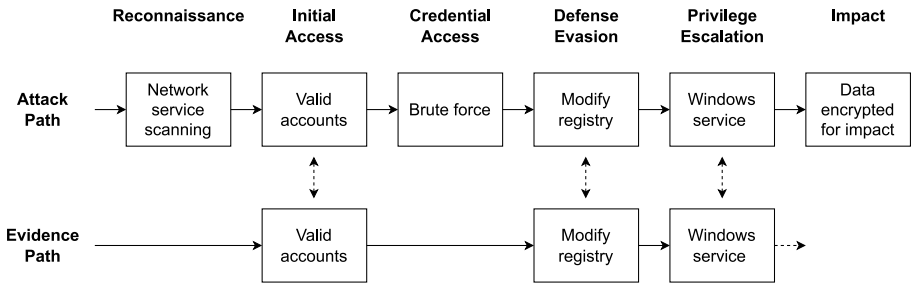


Fig. 7. An example of how path matching works for specific attack and evidence paths. Names of kill chain phases are above the paths.

Algorithm 3: Check Sequence in Path

Input : sequence, attack_path, ip_address, threshold = 0
Output: found_sequence

```
1  sequence_index, path_index, last_successful_path_index ← 0;
2  sequence_length ← length(sequence);
3  attack_path_length ← length(attack_path);
4  found_sequence ← [ ];
5  if ip_address ∈ attack_path then
6      if "lateral movement" ∈ attack_path then
7          path_index ← index(attack_path, "lateral movement");
8          attack_path_length ←
             length(part_of_attack_path_on_ip(ip_address));
9      end
10     while sequence_index < sequence_length do
11         if sequence[sequence_index]["attack_id"] == attack_path[path_index][
             "attack_id"] then
12             last_successful_path_index ← path_index;
13             found_sequence.append(sequence[sequence_index]);
14             sequence_index ← sequence_index + 1;
15         end
16         else path_index ← path_index + 1;
17         if path_index ≥ attack_path_length then
18             path_index ← last_successful_path_index;
19             sequence_index ← sequence_index + 1;
20         end
21     end
22 end
23 matching_score ← length(found_sequence) / length(sequence);
24 if matching_score < threshold then
25     return [ ];
26 end
27 return found_sequence;
```

that determine their starting techniques on Lines 6–9 in Algorithm 3. Therefore, the evidence paths are mapped to individual parts from attack paths. Each part is processed the same way as a whole attack path without lateral movement. For example, Fig. 5 provides three attack paths when only attack techniques are considered. Each of them has two parts divided according to the lateral movement. The matching of the evidence path from Fig. 7 with any of the three second parts would provide a match score of approximately 67%.

The final evaluation for each IP address is determined according to the techniques from the evidence path and criticality of assets. In this phase, we traverse through the alerts found during matching. Hence, the most severe level will often be assigned to the last technique the attacker applies during a multi-step attack. Table 2 shows the evaluation of the evidence path techniques from Fig. 7. The final severity level (Level 3) conforms to the intuition that, in this case, it causes an immediate danger of an incident that will target the organization's objectives.

Table 2
Severity levels assigned to evidence path techniques according to tactics and asset criticality.

Technique	Tactic	Asset	Level
Valid Accounts	Initial Access	Noncritical	4
Modify Registry	Defense Evasion	Noncritical	3
Windows Service	Privilege Escalation	Noncritical	3

6. Evaluation

The evaluation aims to answer whether the proposed approach for triage of evidence paths (i.e., sequences of detection alerts) leads to a realistic estimation of ongoing severe incidents. The critical aspects that must be evaluated are whether the proposed approach filters negative evidence paths and whether identifying severity levels throughout the time respects executed attack techniques. Another aspect is the performance of the proposed approach concerning the running time and size of generated graphs.

6.1. Testbed for evaluation

The testbed contained Windows and Ubuntu hosts, depicted in Fig. 8. The web segment was used for web servers accessible from the Internet, DMZ denoted demilitarized zone, and the internal segment contained hosts that received firewalled traffic. Other hosts included a DNS server and an active directory server controlling the Windows domain containing hosts from the internal segment.

Wazuh SIEM stored host-based data and security alerts from all devices. We used all logs captured by Wazuh, including events not decoded by the decoder. They were accessible using the *wazuh-archives* index pattern in the Wazuh dashboard. MITRE CALDERA of version 4 was used to execute attack procedures on hosts from internal, web, and DMZ segments. These hosts contained deployed CALDERA agents.

Testbed produced no user traffic except for the administrator's actions to maintain systems. However, the testbed was connected to the Internet and ran autonomously. The absence of users was not a disadvantage because the communication of hosts, agents, and host-specific processes caused the background noise. As a result, it provided enough false positives for testing the approach. Adding real users could make the evaluation more complex, add more false positives, and cause the quality of detection methods to be evaluated instead of the proposed approach.

6.2. Attack scenarios

We executed five attack scenarios using the MITRE CALDERA. Two targeted hosts with the Windows operating system and the remaining three Ubuntu hosts. Windows scenarios contained eight attack procedures, while Ubuntu scenarios had six procedures. In addition, the first Ubuntu scenario contained lateral movement between two Ubuntu hosts. The scenarios contained various attack procedures, such as obtaining credentials via different methods, interacting with system

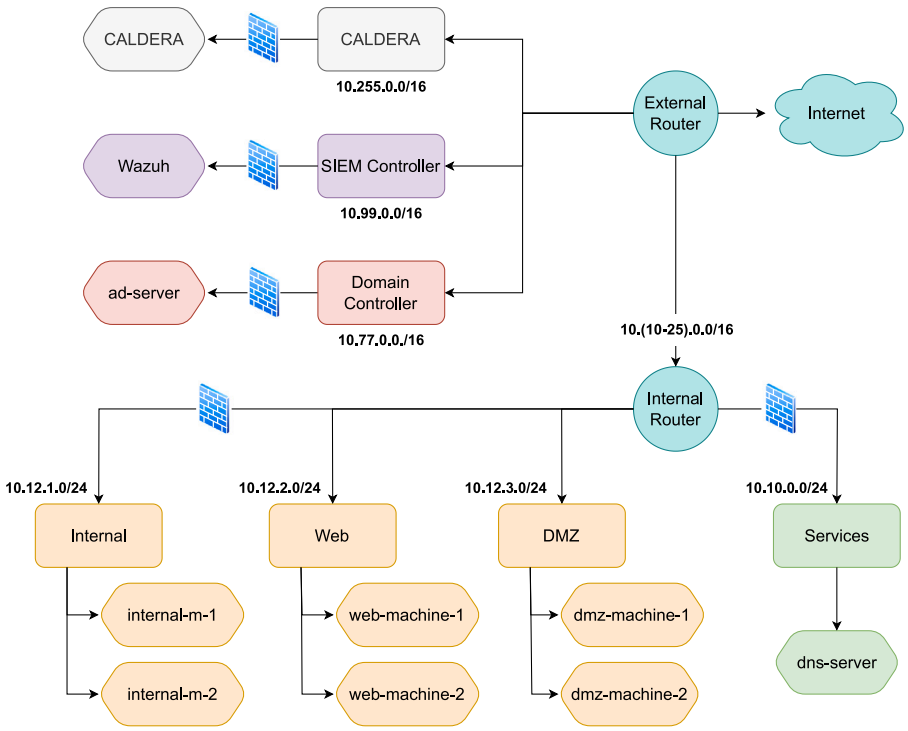


Fig. 8. Network topology for the evaluation hosts depicted by hexagons and network segments by rectangles. Hosts from the internal subnet were under the oversight of the active directory server (ad-server), and all hosts except for CALDERA provided data to Wazuh.

Table 3
Ubuntu scenario for web-machine-2 from Fig. 8. ATT&CK IDs and names of operations were obtained from CALDERA. Only one of the possible tactics is listed for attack techniques. Names of CALDERA operations were slightly edited for brevity.

Tactic	Technique ID	CALDERA operation
Execution	T1059.004	Create and execute bash shell script
Privilege Escalation	T1078.003	Create local account (Linux)
Defense Evasion	T1548.001	Set a setGID flag on file
Discovery	T1033	System owner/user discovery
Collection	T1560.001	Data Compressed – gzip single file
Exfiltration	T1048.002	Exfiltrate data using curl

services, installing software, creating accounts, and deleting data. Each scenario ended with impact or exfiltration techniques.

We have chosen the attack procedures out of the default CALDERA abilities so that they instantiated attack techniques belonging to almost all ATT&CK tactics. Reconnaissance and resource development tactics were not used because each attack in CALDERA already starts on an internal host. The same holds for initial access, even though it is one of the possible tactics for attack technique *T1078 — Valid Accounts*. A scenario for a host from the web segment is listed in Table 3.

Attack procedures were executed during four days. Each procedure was executed with a time gap before the following procedure, so each scenario lasted at least three days. As a result, the data contained a random order of benign and malicious alerts. The attack procedures were ordered according to ATT&CK tactics (i.e., kill chain phases) prior to the execution of scenarios. Scenarios contained 32 different attack procedures, belonging to 31 different ATT&CK techniques. Lists of these manually executed attack techniques represented the ground truth (see supplementary materials [78]).

6.3. Input data

We used captured data in Wazuh from one day before to one day after the evaluation to consider also any activities before and after the scenarios were executed. All Wazuh data from eight machines in Fig. 8 have 8.6 GB and consist of several types of alerts and log events. Many were caused by ordinary processes and hosts' communication in the testbed. The first kind contains alerts of detected ATT&CK techniques. There were approximately 23,700 of them. The second type contains 8 million Suricata alerts that contain essential properties of suspicious IP flows. The last type consists of almost 30,000 remaining alerts generated by system components and applications, e.g., systemd.

There were three categories of true positives — directly detected techniques, detected similar techniques, and techniques detected by custom detection. These three categories contained approximately equal counts of techniques out of 31 ATT&CK techniques from scenarios. An example of a similar technique is *T1569.002 — Service Execution*, which was detected as *T1543.003 — Windows Service*. An example of custom detection is *T1489 — Service Stop* assigned to an unexpected termination of a print spooler on a Windows host. Twelve techniques were not detected with ATT&CK IDs, e.g., the discovery of system owner and users from the command line. In total, 24 alerts became true positives, representing less than 1% of the obtained data.

We prepared generation rules for all detected techniques and other essential ATT&CK techniques to create attack paths that could contain false positive alerts. In general, we defined predicates that expressed network access between hosts, accounts, network services, installed programs, and countermeasures to attack techniques. Attack goals were determined according to the positions of hosts in the testbed. Internal hosts, the DNS server, and the active directory server were marked as critical hosts for evaluation. In contrast, hosts from web and DMZ segments were considered noncritical due to their function and accessibility from the Internet. The input also contained information about which countermeasures were not applied in the testbed. The ruleset can be found in supplementary materials [78]. We reused some of the already published rules at [85].

6.4. Evidence paths and KCAG

Communication alerts by Suricata and ATT&CK alerts were used to construct evidence paths and, with other log events, the kill chain

Table 4
How many log events (including logged ATT&CK and communication alerts) appeared in the data, and aggregated events were identified for hosts and scenarios. Percentage was computed for aggregated events.

Hostname	Scenario	Events	Aggregated	Percentage
internal-m-1	Windows 1	1126	474	< 0.1%
internal-m-2	Windows 2	1132	501	< 0.1%
web-machine-1	Ubuntu 1	1,012,409	9529	1.0%
web-machine-2	Ubuntu 3	1,012,840	8601	0.9%
dmz-machine-1	Ubuntu 1	857,851	206,343	21.3%
dmz-machine-2	Ubuntu 2	858,352	206,871	21.4%
ad-server	Windows 2	22,236	840	0.1%
dns-server	–	4,427,580	535,616	55.3%

Table 5
Data size and performance of generating KCAGs with and without ports. Concerning the time to list paths with ports, we did not wait until it was fully finished.

	With ports	Without ports
Input facts	270	178
Vertices in KCAG	22,273	3955
Edges in KCAG	118,642	11,145
Time to generate a file	≈ 1 min	≈ 1 min
Time to generate a graph	≈ 0.5 min	≈ 10 s
Time to list paths of length up to 10	> 2 h	≈ 20 s

attack graph. Table 4 contains counts of events for hostnames and CALDERA scenarios that influenced them. A higher count for Ubuntu hosts was caused by Suricata warnings about stream packets that provided information about communicating IP addresses.

Our custom aggregating functionality reduced the count of events to 12% of the previous value on average by adding the count of repeating events during a time interval to aggregated ones. As a result, we had 475 valid evidence paths respecting kill chain phases for all hosts. Each evidence path had two to nine alerts, and almost 80% consisted of four to seven. Five paths included the lateral movement tactic at their beginning to deal with the possibility of lateral movement during path matching. Most events with ATT&CK IDs from aggregated sequences (including benign actions) appeared in evidence paths.

The proposed approach generates KCAG without port numbers. The advantage of this adjustment is visible in Table 5. Using port numbers resulted in a larger attack graph, more input facts, and a longer execution time. Determining all paths with lengths up to ten vertices took significantly longer, justifying the correctness of ignoring port numbers. Even though the KCAG without port numbers contains fewer attack paths, all are meaningful when specific port numbers (i.e., network services) allow these techniques to be executed. Time aspects were evaluated using a desktop computer with 64 GB RAM, 16 CPU cores, and a processor's clock speed of 2.5 GHz. However, KCAGs were generated on a virtual machine on the same computer. One KCAG path without port numbers is depicted in Fig. 9.

Approximately half of ATT&CK IDs from detection alerts appeared in the attack paths. On the contrary, only one-third of ATT&CK IDs from the attack paths were also present in Wazuh. The generated KCAG from the network topology description contained lateral movements, making the quick and exhaustive enumeration of all paths impossible. However, we tested that using attack paths with up to ten vertices and five attack techniques was feasible during evaluation and could be executed in less than a minute. It relies on creating the longest possible evidence paths and their partial matching with attack paths. Moreover, the length of evidence paths has an upper bound in practice. Twelve MITRE ATT&CK tactics appear in alerts, and not all are usually used during one multi-step cyber attack. If longer attack paths are necessary, we should not list all of them but instead use the breadth-first search to compare alerts with attack techniques continuously.

6.5. Results of path matching

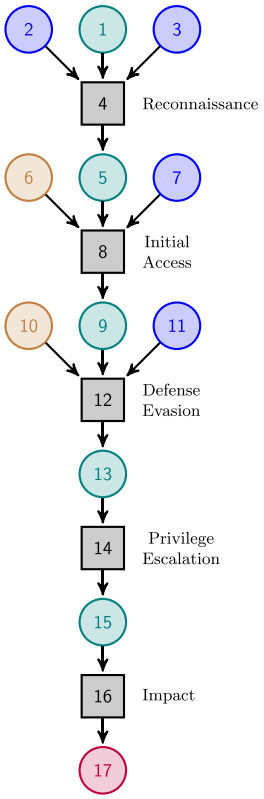
A *positive evidence path* is a sequence containing only techniques appearing in true positive alerts *sorted in the correct order*. The correct order is determined by manually executed scenarios (i.e., the ground truth), from which some attack steps are detected as alerts. An exception from this definition holds for compulsory initial access or lateral movement techniques at the beginning of evidence paths for which we did not require to be true positives. Attack scenarios in CALDERA did not start with initial access tactics because the agents were already installed on hosts.

We identified ten positive evidence paths out of 475 evidence paths when no threshold during path matching is applied. These positive evidence paths represent 2.1% of evidence paths. However, even when a threshold is not used, creating evidence paths filters many combinations of alerts because it creates only sequences that respect kill chain phases out of more than 20,000 detected alerts. Consequently, we assigned a match score to each evidence path and observed what match score was assigned to each positive evidence path. Results are comprehensively described by the Receiver Operating Characteristics (ROC) curve in Fig. 10. This ROC curve describes the relationship between true positive and false positive rates. Area Under ROC Curve (AUC) has a value of 0.95, considered a very good performance. In other words, positive evidence paths usually obtain higher match scores than negative ones.

As a next step, we experimented with different threshold levels of match scores to classify results. We interpreted our approach as a classifier that divides evidence paths based on the threshold into positive and negative evidence paths. A threshold of 0.5 was able to correctly classify all positive evidence paths while wrongly classifying 41.5% of negative evidence paths. The result of having 8 out of 10 correctly classified positive evidence paths was compensated by approximately 6.2% of wrongly classified negative paths for a threshold value equal to $\frac{2}{3}$. A threshold of 0.75 provided 7 out of 10 correctly classified positive evidence paths, compensated by approximately 2.6% of wrongly classified negative paths. Higher thresholds caused that half of the positive evidence paths and almost all negative paths were classified correctly. Therefore, we postulate that practically applicable thresholds would be around 0.7. At the same time, use cases when all positive evidence paths are required could use the threshold around 0.5 for KCAG paths of length up to ten vertices.

Multiclass analysis of results revealed results from confusion matrices in Table 6 for thresholds of zero and $\frac{2}{3}$. Predicted levels are values the approach outputted, and expected levels contain values the approach should output if no threshold is used. Confusion matrices compare predicted severity levels for the most severe alerts from evidence paths concerning expected severity levels based on true positive alerts. The confusion matrices express a different aspect than the previous paragraph that strictly compared overall evidence paths with executed attack scenarios. Therefore, results for the zero threshold are the most illustrative for the classification operation, while the previous paragraph measured the success of using the threshold for overall evidence paths.

Results show that when the approach wrongly classifies positive and negative evidence paths for the threshold of zero, it can still classify them to correct severity levels since their alerts are ordered, different techniques can belong to the same tactic, and several tactics belong to specific levels in Table 1. However, even though it happens for cases from diagonals of Table 6, it is not a general rule. The higher threshold is used, the less evidence paths are classified to the most severe Levels 1 and 2, and the more evidence paths are classified to the default severity Level 5 since they do not pass over the threshold. We should also note that confusion matrices for thresholds had no expected Levels 4 because the approach creates the longest possible evidence paths. Moreover, Level 4 represents only initial access, and the evaluation data had few such techniques. Single alert initial access evidence paths would belong



ID	Value
1	externalActor(*10_255_1_37*).
2	networkService(*10_12_1_10*, _, _, _).
3	accessAllowed(*10_255_1_37*, *10_12_1_10*, _, _).
4	T1595 - Active Scanning
5	networkConnection(*2*, *authentication*, *10_12_1_10*, _, _).
6	passwordPolicies(*no*).
7	hasAccount(_, *user*, *10_12_1_10*, *windows*).
8	T1078 - Valid Accounts
9	account(*2*, *authentication*, *user*, _, *10_12_1_10*, *windows*).
10	restrictRegistryPermissions(*no*).
11	installed(*10_12_1_10*, *windows*).
12	T1112 - Modify Registry
13	file(*2*, *integrity*, *10_12_1_10*, *registry*).
14	T1543.003 - Windows Service
15	system(*2*, *authorization*, *10_12_1_10*, *root*).
16	T1529 - System Shutdown / Reboot
17	system(*2*, *availability*, *10_12_1_10*, _).

Fig. 9. Attack path excerpt from KCAG generated during evaluation. The left part visualizes relationships between vertices and kill chain phases. The right table contains descriptions of vertices. CALDERA machine had an IP address of 10.255.1.37, while an internal machine had 10.12.1.10.

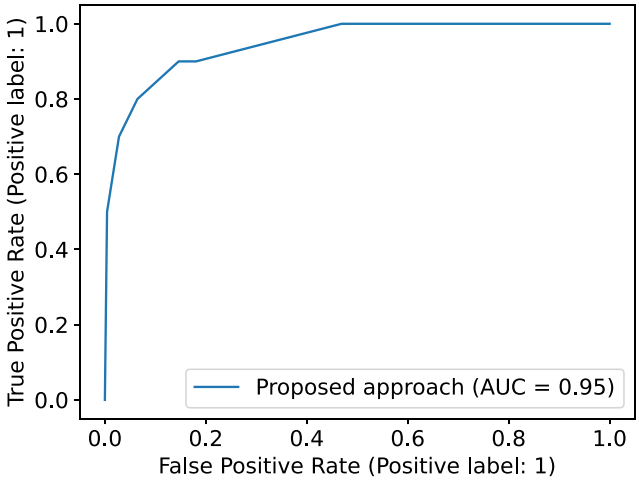


Fig. 10. ROC curve for results obtained from match scores of evidence paths compared with the ground truth.

Table 6
Confusion matrices for severity levels when thresholds of zero and $\frac{2}{3}$ were used. Expected levels represent values that should be outputted based on true positive alerts.

		Predicted levels ($t = 0$)					Predicted levels ($t = \frac{2}{3}$)				
		1	2	3	4	5	1	2	3	4	5
Expected Levels	1	230	21	11	0	0	8	0	0	0	254
	2	7	183	10	0	0	0	29	0	0	171
	3	0	0	13	0	0	0	0	0	0	13
	4	0	0	0	0	0	0	0	0	0	0
	5	0	0	0	0	0	0	0	0	0	0

there. Accuracy and micro-averages of precision and recall are equal to 0.90, the macro-average of precision to 0.75, and the macro-average of recall to 0.93 for threshold of zero.

Evaluation of whether outputted severity levels according to the proposed escalation ladder respect the executed techniques in scenarios is summarized in Table 7. The ground truth was determined as the severity levels for the most severe techniques from scenarios detected in the captured data. We did not evaluate detection functionality but whether the approach chose the correct techniques. If all techniques were detected, all hosts from web and DMZ segments would contain severity Level 2 in the ground truth. Technique T1548.003 is present for three IP addresses in Table 7 since we used two different CALDERA procedures that mapped to it.

We discovered that all severity levels for hosts from internal, web, and DMZ segments from Fig. 8 were achieved. The difference was that the escalation approach sometimes added false positive techniques on the same level when no threshold was used. In other words, the approach did not output wrong severity levels, but false positive alerts were added to the list of alerts that belong to this severity level.

AD and DNS servers differed from the first six hosts since they were not the primary targets. Table 7 shows that the AD server obtained severity Level 1 instead of 2 because the brute force of the domain user in one Windows scenario was also detected as false positive account access removal when too many unsuccessful login attempts were observed. Despite false positive alerts, the DNS server had no valid evidence path.

The progress of the computation when no threshold was used is listed in Table 8. We consistently executed the approach for data from the first day to the day denoted as D_i, e.g., the D_3 column was computed based on data captured during the first three days. The table clearly shows the impact on severity levels caused by false positive alerts. These alerts cause the assignment of higher severity levels. However, SOC analysts could resolve these alerts and decrease severity levels again to obtain the correct levels, mainly when the

Table 7
The ground truth severity levels for hosts considered in the evaluation and detected attack techniques. The last column expresses whether the techniques were present in the output when no threshold for match score was used.

Hostname	ATT&CK	Name of technique	Level	Result
internal-m-1	T1496	Resource Hijacking	1	✓
internal-m-2	T1489	Service Stop	1	✓
web-machine-1	T1529	System Shutdown/Reboot	2	✓
web-machine-2	T1548.003	Sudo and Sudo Caching	3	✓
dmz-machine-1	T1548.003	Sudo and Sudo Caching	3	✓
dmz-machine-2	T1548.003	Sudo and Sudo Caching	3	✓
ad-server	T1110	Brute Force	2	×
dns-server	–	–	–	✓

Table 8
Progress of severity levels computation without any threshold. D_i denotes cumulative results from the proposed approach after day number i in the dataset. In contrast, G_i denotes levels after day number i for detected alerts from the ground truth scenarios. Asterisks denote when the ground truth techniques were present in the output. D_6 and G_6 were equal to D_5 and G_5 respectively.

Hostname	D_1	D_2	D_3	D_4	D_5	G_1	G_2	G_3	G_4	G_5
internal-m-1	1	1	1	1	1*	–	2	2	2	1
internal-m-2	1	1	1	1	1*	–	–	2	2	1
web-machine-1	3	3	3	2*	2*	–	–	–	2	2
web-machine-2	3	3	3*	3*	3*	–	–	3	3	3
dmz-machine-1	3	3*	3*	3*	3*	–	3	3	3	3
dmz-machine-2	–	3*	3*	3*	3*	–	3	3	3	3

Table 9
Progress of severity levels computation with a threshold of 0.4. The meaning of D_i and asterisks is the same as in Table 8. The values should be compared with the ground truth levels in Table 8.

Hostname	D_1	D_2	D_3	D_4	D_5
internal-m-1	2	2*	2*	2*	1*
internal-m-2	2	2	2*	2*	1*
web-machine-1	3	3	3	–	–
web-machine-2	3	3	3*	3*	3*
dmz-machine-1	3	–	–	–	–
dmz-machine-2	–	3*	3*	3*	3*

method outputs only a low amount of false positives. Table 8 shows that even benign actions caused alerts during D_1 that change severity levels to the worst values.

Using value 0.4 as a threshold resulted in having all severity levels correct except for two IP addresses from the lateral movement scenario (see Table 9). Agents on the two IP addresses could observe only parts of the scenario because lateral movement led to another host. This result is meaningful since the two IP addresses could appear in shorter evidence paths. The threshold also improved results for Windows hosts, while some false positive alerts of all hosts remained for days when no scenarios were executed. Higher threshold values than 0.4 caused only Windows scenarios to be classified correctly. Again, it is a meaningful result concerning only one-third of attack techniques detected from scenarios for web and DMZ hosts.

6.6. Discussion of limitations and advantages

The proposed approach uses attack graphs prone to scalability issues [32]. The issues would appear for networks containing hundreds of hosts and tens of vulnerabilities per host [32]. In our case, we cannot influence the count of hosts but could address CVEs. A partial solution to the significant increase in the count of attack paths would be to deal with categories of CVEs instead of specific CVEs. It would mean that we consider only one general vulnerability representing a group of CVEs on one host with the same impact. It would be as if we merge the group of vulnerabilities into one vertex. Since they have the same impact, they are connected to the same vertices in KCAG. Impacts of CVEs are already used in the proof-of-concept implementation to determine attack techniques that exploit vulnerabilities, but merging is not implemented in the current state. It can be done in the future work.

Cycles represent the next challenge for attack graphs [32]. In this paper, we encounter the possibility of the attacker returning to already visited hosts using lateral movements. Prioritizing attacks with lateral movement would be more complicated with the increased number of hosts since many possible tuples of devices could be jeopardized in such attacks. However, evidence paths contain only parts of the multi-step attacks executed on individual hosts. Therefore, KCAGs are more prone to this issue. According to MulVAL's source code, it generates attack graphs with cycles but does not get stuck in an infinite loop. Post-processing its attack graphs only requires considering simple paths to avoid the problem.

A practical disadvantage is that MulVAL uses logic programming that requires specifying the correct predicates for all rules of attack techniques. Compared with other programming paradigms, such as object-oriented programming, maintaining a hierarchy of objects from data models is more labor-intensive since the generator cannot automatically determine all substitutions of equivalent classes (e.g., obtained password for SSH service and user privileges on a system) unless the generation rules exhaustively list them. Another limitation of the approach is that we always processed the amount of data stored in operation memory, and we did not store timestamps because the time window was short.

A limitation of evaluation is the size of the testbed. The testbed was created to be comparable with the related work (see Section 3.2) since it used ten hosts and five attack scenarios with 31 ATT&CK techniques. It also contained variability of operating systems and provided various kinds of data — syslog and Windows log events, IP flows or IDS alerts, and compulsory XDR alerts. Creating a suitable testbed for evaluating threat modeling approaches is a future work.

Results show that the approach can filter a lot of false positives and identify the correct evidence paths worth attention (see Fig. 10). Tables 8 and 9 show that even though the approach identifies a set of malicious evidence paths containing ongoing incidents, there are better strategies than focusing on one (the most severe) path out of several evidence paths in practice. However, an advantage may be accompanying severity levels by alerts or whole evidence paths. The length of evidence paths could provide another dimension that can be considered by SOC analysts or crisis management. Moreover, a practical advantage of the approach is that false positive evidence paths are also usable for improving security posture. They reveal to security analysts that adversaries could employ these techniques in the future, but they have not yet. Right now, a possible attack path in the network and an accidental evidence path of false positive alerts exist.

Results from the evaluation are comparable to related work in several aspects. The data we used contained only a low fraction of alerts among log events, similar to [51]. Provenance graphs are generated in seconds in [9] as KCAGs from this paper. However, their use cases differ because provenance graphs depict data provenance operations. [9] used three attack campaigns for evaluation, obtained five true and 676 false tactical provenance graphs, and 99% AUC. In our case, ten out of 475 evidence paths were positive for five attack scenarios, resulting in 95% AUC. Our approach does not provide such a perfect division of sequences of alerts to obtain the same AUC.

7. Conclusion

In this paper, we proposed an approach for severity-based incident triage, which can recommend evidence paths (i.e., sequences of detection alerts) that represent executed multi-step attacks. We used the kill chain attack graphs created using observed hosts' logs and network IP flows. Kill chain phases were also used to create evidence paths containing MITRE ATT&CK techniques. The approach identifies the most advanced kill chain phase for one of the incidents indicated by alerts from found paths. It assigns severity levels to each found evidence

path according to the most severe phase and the criticality of assets that are jeopardized by it. In such a way, we can triage incidents based on their severity. SOC analysts can further handle the incidents and escalate them to other analysts or crisis management.

We evaluated the proposed approach concerning the realistic estimation of executed attack scenarios. The evaluation showed that the approach assigns correct severity during the triage of evidence paths and identifies ongoing severe incidents indicated by security alerts. The approach achieves 0.95 AUC during exhaustive classification with all created evidence paths from detected alerts. Even though severity levels determined by the approach very often correspond to the ground truth, we also observed two cases when results did not equal it. These cases were the lateral movement that split the scenario into two parts for two hosts and a low amount of detected techniques for high thresholds. We released supplementary materials that can be used to reproduce the results. They contain open-source implementation of the CTS Analyzer, evaluation scripts, and data captured during the attack scenarios [78].

Our approach and related work indicate that focusing on sequences of detection alerts instead of individual alerts is a promising research area for incident triage. These approaches could have the role of a recommender system for SOC analysts and crisis management in the future. They would relieve analysts from repetitive manual work that causes alert fatigue. However, the role and skills of analysts for proactive cyber threat hunting will still be necessary because sophisticated attackers can execute their actions without being detected.

CRediT authorship contribution statement

Lukáš Sadlek: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Data curation, Conceptualization. **Muhammad Mudassar Yamin:** Writing – review & editing, Supervision, Resources, Methodology, Conceptualization. **Pavel Čeleda:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Basel Katt:** Writing – review & editing, Conceptualization.

Declaration of competing interest

The authors declare no conflict of interest.

Acknowledgments

Muhammad Mudassar Yamin and Basel Katt were supported by Norwegian Research Council ASCERT project No. 329062. We would also like to thank Md Mujahid Islam Peal for his valuable help with preparing the infrastructure for evaluation.

Data availability

The data are available in supplementary materials referenced using a URL link.

References

[1] Wang X, Yang X, Liang X, Zhang X, Zhang W, Gong X. Combating alert fatigue with AlertPro: Context-aware alert prioritization using reinforcement learning for multi-step attack detection. Comput Secur 2024;137:103583. http://dx.doi.org/10.1016/j.cose.2023.103583.

[2] Hassan WU, Guo S, Li D, Chen Z, Jee K, Li Z, Bates A. Nodoze: Combatting threat alert fatigue with automated provenance triage. In: Network and distributed systems security symposium. 2019, p. 15, URL https://par.nsf.gov/biblio/10085663.

[3] Orca Security. 2022 cloud security alert fatigue report. 2022, URL https://orca.security/wp-content/uploads/2022/03/Orca-2022-Cloud-Security-Alert-Fatigue-Report.pdf (Accessed on 6 May 2024).

[4] IDC Research, Inc. In cybersecurity every alert matters. 2021, URL https://www.criticalstart.com/resources/in-cybersecurity-every-alert-matters/ (Accessed on 6 May 2024).

[5] Ahmad A, Maynard SB, Desouza KC, Kotsias J, Whitty MT, Baskerville RL. How can organizations develop situation awareness for incident response: A case study of management practice. Comput Secur 2021;101:102122. http://dx.doi.org/10.1016/j.cose.2020.102122.

[6] Ross R, Pillitteri V, Graubart R, Bodeau D, McQuaid R. Developing cyber-resilient systems: A systems security engineering approach. NIST Special Publication 800-160, Volume 2, Revision 1, Gaithersburg, Maryland, USA: National Institute of Standards and Technology; 2021, http://dx.doi.org/10.6028/NIST.SP.800-160v2r1.

[7] Trimintzios P, Holfeldt R, Koraeus M, Uckan B, Gavrila R, Makrodimitris G. Report on cyber crisis cooperation and management: Comparative study on the cyber crisis management and the general crisis management. European Network and Information Security Agency; 2015, http://dx.doi.org/10.2824/34669.

[8] Kotenko I, Gaifulina D, Zelichenok I. Systematic literature review of security event correlation methods. IEEE Access 2022;10:43387–420. http://dx.doi.org/10.1109/ACCESS.2022.3168976.

[9] Hassan WU, Bates A, Marino D. Tactical provenance analysis for endpoint detection and response systems. In: 2020 IEEE symposium on security and privacy. SP, 2020, p. 1172–89. http://dx.doi.org/10.1109/SP40000.2020.00096.

[10] Hu H, Liu J, Zhang Y, Liu Y, Xu X, Tan J. Attack scenario reconstruction approach using attack graph and alert data mining. J Inf Secur Appl 2020;54:102522. http://dx.doi.org/10.1016/j.jisa.2020.102522.

[11] Nadeem A, Verwer S, Moskal S, Yang SJ. Alert-Driven Attack Graph Generation Using S-PDFA. IEEE Trans Depend Secur Comput 2022;19(2):731–46. http://dx.doi.org/10.1109/TDSC.2021.3117348.

[12] Crowley C, Pescatore J. A SANS 2021 survey: Security operations center (SOC). 2021, URL https://www.sans.org/white-papers/sans-2021-survey-security-operations-center-soc/ (Accessed on 6 May 2024).

[13] Alahmadi BA, Axon L, Martinovic I. 99% false positives: A qualitative study of SOC analysts' perspectives on security alarms. In: 31st USENIX security symposium (USENIX security 22). Boston, MA: USENIX Association; 2022, p. 2783–800, URL https://www.usenix.org/conference/usenixsecurity22/presentation/alahmadi.

[14] Sadlek L, Čeleda P, Tovarňák D. Identification of attack paths using kill chain and attack graphs. In: NOMS 2022-2022 IEEE/IFIP network operations and management symposium. 2022, p. 1–6. http://dx.doi.org/10.1109/NOMS54207.2022.9789803.

[15] The MITRE Corporation. MITRE ATT&CK. 2015-2024, URL https://attack.mitre.org/ (Accessed on 6 May 2024).

[16] FIRST - Forum of Incident Response and Security Teams. Computer security incident response team (CSIRT) services framework. 2015-2024, URL https://www.first.org/standards/frameworks/csirts/csirt_services_framework_v2.1 (Accessed on 6 May 2024).

[17] Maj M, Reijers R, Stikvoort D. Good practice guide for incident management. 2010, URL https://www.enisa.europa.eu/publications/good-practice-guide-for-incident-management.

[18] Portesi S, De Muynck J. Strategies for incident response and cyber crisis cooperation. European Network and Information Security Agency; 2016, http://dx.doi.org/10.2824/967546.

[19] Østby G, Lovell KN, Katt B. EXCON teams in cyber security training. In: 2019 international conference on computational science and computational intelligence. CSCI, IEEE; 2019, p. 14–9. http://dx.doi.org/10.1109/CSCI49370.2019.00010.

[20] Østby G, Katt B. Cyber crisis management roles – a municipality responsibility case study. In: Murayama Y, Velev D, Zlateva P, editors. Information technology in disaster risk reduction. Cham: Springer International Publishing; 2020, p. 168–81. http://dx.doi.org/10.1007/978-3-030-48939-7_15.

[21] National Institute of Standards and Technology. Framework for improving critical infrastructure cybersecurity. 2018, http://dx.doi.org/10.6028/NIST.CSWP.04162018, Accessed on 6 May 2024.

[22] Muniz J, McIntyre G, AlFardan N. Security operations center: Building, operating, and maintaining your SOC. Indianapolis (USA): Cisco Press; 2016.

[23] Gartner, Inc. Security orchestration, automation and response (SOAR). 2024, URL https://www.gartner.com/en/information-technology/glossary/security-orchestration-automation-response-soar (Accessed on 6 May 2024).

[24] Kokulu FB, Soneji A, Bao T, Shoshitaishvili Y, Zhao Z, Doupé A, Ahn G-J. Matched and mismatched SOCs: A qualitative study on security operations center issues. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. CCS '19, New York, NY, USA: Association for Computing Machinery; 2019, p. 1955–70. http://dx.doi.org/10.1145/3319535.3354239.

[25] Bianco D. The pyramid of pain. 2013, URL http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html (Accessed on 6 May 2024).

[26] Stillions R. The DML model. 2014, URL http://ryanstillions.blogspot.com/2014/04/the-dml-model_21.html (Accessed on 6 May 2024).

[27] Kostyuk N, Powell S, Skach M. Determinants of the cyber escalation ladder. Cyber Déf Rev 2018;3(1):123–34, URL https://www.jstor.org/stable/26427380.

[28] Sagan SD. Nuclear alerts and crisis management. Int Secur 1985;9(4):99–139. http://dx.doi.org/10.2307/2538543, (Accessed on 13 Sep 2024).

[29] Hutchins EM, Cloppert MJ, Amin RM. Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains. Lead Issues Inf Warf Secur Res 2011;1(1):80–106.

[30] Yadav T, Rao AM. Technical aspects of cyber kill chain. In: Abawajy JH, Mukherjea S, Thampi SM, Ruiz-Martínez A, editors. Security in computing and communications. Cham: Springer International Publishing; 2015, p. 438–52. http://dx.doi.org/10.1007/978-3-319-22915-7_40.
[31] Pols P. The unified kill chain, URL https://www.unifiedkillchain.com/assets/The-Unified-Kill-Chain.pdf (Accessed on 6 May 2024).
[32] Kaynar K. A taxonomy for attack graph generation and usage in network security. J Inf Secur Appl 2016;29:27–56. http://dx.doi.org/10.1016/j.jisa.2016.02.001.
[33] Ghosh N, Chokshi I, Sarkar M, Ghosh SK, Kaushik AK, Das SK. NetSecuritas: An integrated attack graph-based security assessment tool for enterprise networks. In: Proceedings of the 2015 international conference on distributed computing and networking. ICDCN '15, New York, NY, USA: Association for Computing Machinery; 2015, p. 1–10. http://dx.doi.org/10.1145/2684464.2684494.
[34] Yiğit B, Gür G, Alagöz F, Tellenbach B. Cost-aware securing of IoT systems using attack graphs. Ad Hoc Netw 2019;86:23–35. http://dx.doi.org/10.1016/j.adhoc.2018.10.024.
[35] Andreolini M, Colacino VG, Colajanni M, Marchetti M. A framework for the evaluation of trainee performance in cyber range exercises. Mob Netw Appl 2020;25:236–47. http://dx.doi.org/10.1007/s11036-019-01442-0.
[36] Yi S, Peng Y, Xiong Q, Wang T, Dai Z, Gao H, Xu J, Wang J, Xu L. Overview on attack graph generation and visualization technology. In: 2013 international conference on anti-counterfeiting, security and identification. ASID, 2013, p. 1–6. http://dx.doi.org/10.1109/ICASID.2013.6825274.
[37] Ou X, Govindavajhala S, Appel AW. Mulval: A logic-based network security analyzer. In: USeNIX security symposium. vol. 8, Baltimore, MD; 2005, p. 113–28, URL https://www.usenix.org/legacy/event/sec05/tech/full_papers/ou/ou_html/.
[38] Jajodia S, Noel S, Kalapa P, Albanese M, Williams J. Cauldron mission-centric cyber situational awareness with defense in depth. In: 2011 - MILCOM 2011 military communications conference. 2011, p. 1339–44. http://dx.doi.org/10.1109/MILCOM.2011.6127490.
[39] Bopche GS, Mehtre BM. Attack graph generation, visualization and analysis: Issues and challenges. In: Mauri JL, Thampi SM, Rawat DB, Jin D, editors. Security in computing and communications. Berlin, Heidelberg: Springer Berlin Heidelberg; 2014, p. 379–90. http://dx.doi.org/10.1007/978-3-662-44966-0_37.
[40] CVE Website, The MITRE Corporation (1999-2024) URL https://www.cve.org/ (Accessed on 6 May 2024).
[41] Sadlek L, Čeleda P, Tovarňák D. Current Challenges of Cyber Threat and Vulnerability Identification Using Public Enumerations. In: The 17th international conference on availability, reliability and security (ARES 2022). New York, NY, USA: ACM; 2022, http://dx.doi.org/10.1145/3538969.3544458, 8 pages.
[42] The MITRE Corporation. MITRE CALDERA, URL https://caldera.mitre.org/ (Accessed on 6 May 2024).
[43] Hofstede R, Čeleda P, Trammell B, Drago I, Sadre R, Sperotto A, Pras A. Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX. IEEE Commun Surv Tutor 2014;16(4):2037–64. http://dx.doi.org/10.1109/COMST.2014.2321898.
[44] Netflix, Inc.. Dispatch – documentation. 2024, URL https://netflix.github.io/dispatch/ (Accessed on 6 May 2024).
[45] Wazuh, Inc. Wazuh: The Open Source Security Platform. 2024, URL https://wazuh.com/ (Accessed on 6 May 2024).
[46] Microsoft Corporation. Sysmon v15.14. 2024, URL https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon (Accessed on 6 May 2024).
[47] Gerhards R. The syslog protocol. In: Request for comments, (5424). RFC Editor; 2009, http://dx.doi.org/10.17487/RFC5424.
[48] Open Information Security Foundation (OISF). Suricata. 2024, URL https://suricata.io/ (Accessed on 6 May 2024).
[49] Wazuh, Inc. Wazuh documentation. 2024, URL https://documentation.wazuh.com/current/index.html (Accessed on 6 May 2024).
[50] Zhong C, Lin T, Liu P, Yen J, Chen K. A cyber security data triage operation retrieval system. Comput Secur 2018;76:12–31. http://dx.doi.org/10.1016/j.cose.2018.02.011.
[51] Liu Y, Shu X, Sun Y, Jang J, Mittal P. RAPID: Real-Time Alert Investigation with Context-Aware Prioritization for Efficient Threat Discovery. In: Proceedings of the 38th annual computer security applications conference. ACSAC '22, New York, NY, USA: Association for Computing Machinery; 2022, p. 827–40. http://dx.doi.org/10.1145/3564625.3567997.
[52] Bryant BD, Saiedian H. Improving SIEM alert metadata aggregation with a novel kill-chain based classification model. Comput Secur 2020;94:101817. http://dx.doi.org/10.1016/j.cose.2020.101817.
[53] Mouratidis H, Islam S, Santos-Olmo A, Sanchez LE, Ismail UM. Modelling language for cyber security incident handling for critical infrastructures. Comput Secur 2023;128:103139. http://dx.doi.org/10.1016/j.cose.2023.103139.
[54] Liu J, Zhang R, Liu W, Zhang Y, Gu D, Tong M, Wang X, Xue J, Wang H. Context2Vector: Accelerating security event triage via context representation learning. Inf Softw Technol 2022;146:106856. http://dx.doi.org/10.1016/j.infsof.2022.106856.
[55] Aminanto ME, Zhu L, Ban T, Isawa R, Takahashi T, Inoue D. Combating threat-alert fatigue with online anomaly detection using isolation forest. In: Gedeon T, Wong KW, Lee M, editors. Neural information processing. Cham: Springer International Publishing; 2019, p. 756–65. http://dx.doi.org/10.1007/978-3-030-36708-4_62.
[56] Fan S, Wu S, Wang Z, Li Z, Yang J, Liu H, Liu X. ALEAP: Attention-based LSTM with Event Embedding for Attack Projection. In: 2019 IEEE 38th international performance computing and communications conference. IPCCC, 2019, p. 1–8. http://dx.doi.org/10.1109/IPCCC47392.2019.8958761.
[57] Ede Tv, Aghakhani H, Spahn N, Bortolameotti R, Cova M, Continella A, Steen Mv, Peter A, Kruegel C, Vigna G. DEEPCASE: Semi-Supervised Contextual Analysis of Security Events. In: 2022 IEEE symposium on security and privacy. SP, 2022, p. 522–39. http://dx.doi.org/10.1109/SP46214.2022.9833671.
[58] Zhong C, Yen J, Liu P, Erbacher RF. Learning from experts' experience: Toward automated cyber security data triage. IEEE Syst J 2019;13(1):603–14. http://dx.doi.org/10.1109/JSYST.2018.2828832.
[59] Milajerdi SM, Eshete B, Gjomemo R, Venkatakrishnan V. POIROT: Aligning attack behavior with kernel audit records for cyber threat hunting. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security. CCS '19, New York, NY, USA: ACM; 2019, p. 1795–812. http://dx.doi.org/10.1145/3319535.3363217.
[60] Milajerdi SM, Gjomemo R, Eshete B, Sekar R, Venkatakrishnan VN. HOLMES: Real-time APT detection through correlation of suspicious information flows. In: 2019 IEEE symposium on security and privacy. SP, IEEE; 2019, p. 1137–52. http://dx.doi.org/10.1109/SP.2019.00026.
[61] Mohammadzad M, Karimpour J, Mahan F. MAGD: Minimal attack graph generation dynamically in cyber security. Comput Netw 2023;236:110004. http://dx.doi.org/10.1016/j.comnet.2023.110004.
[62] Tian J-w, Li X, Tian Z, Qi W-h. Network attack path reconstruction based on similarity computation. In: 2017 13th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-fSKD). 2017, p. 2457–61. http://dx.doi.org/10.1109/FSKD.2017.8393160.
[63] Alrehaili M, Alshamrani A. An attack scenario reconstruction approach using alerts correlation and a dynamic attack graph. In: 2023 eighth international conference on mobile and secure services (mobiSecServ). CFP23RAC-ART, 2023, p. 1–8. http://dx.doi.org/10.1109/MobiSecServ58080.2023.10329144.
[64] Mao B, Liu J, Lai Y, Sun M. MIF: A multi-step attack scenario reconstruction and attack chains extraction method based on multi-information fusion. Comput Netw 2021;198:108340. http://dx.doi.org/10.1016/j.comnet.2021.108340.
[65] Lyu H, Liu J, Lai Y, Mao B, Huang X. AGCM: A multi-stage attack correlation and scenario reconstruction method based on graph aggregation. Comput Commun 2024;224:302–13. http://dx.doi.org/10.1016/j.comcom.2024.06.016.
[66] Myneni S, Chowdhary A, Sabur A, Sengupta S, Agrawal G, Huang D, Kang M. DAPT 2020 - constructing a benchmark dataset for advanced persistent threats. In: Wang G, Ciptadi A, Ahmadzadeh A, editors. Deployable machine learning for security defense. Cham: Springer International Publishing; 2020, p. 138–63. http://dx.doi.org/10.1007/978-3-030-59621-7_8.
[67] Xiong W, Legrand E, Å berg O, Lagerström R. Cyber security threat modeling based on the MITRE enterprise att&ck matrix. Softw Syst Model 2022;21:157–77. http://dx.doi.org/10.1007/s10270-021-00898-7.
[68] Gylling A, Ekstedt M, Afzal Z, Eliasson P. Mapping cyber threat intelligence to probabilistic attack graphs. In: 2021 IEEE international conference on cyber security and resilience. CSR, 2021, p. 304–11. http://dx.doi.org/10.1109/CSR51186.2021.9527970.
[69] Inokuchi M, Ohta Y, Kinoshita S, Yagyu T, Stan O, Bitton R, Elovici Y, Shabtai A. Design procedure of knowledge base for practical attack graph generation. In: Proceedings of the 2019 ACM Asia conference on computer and communications security. 2019, p. 594–601. http://dx.doi.org/10.1145/3321705.3329853.
[70] Wang Y, Guo Y, Fang C. An end-to-end method for advanced persistent threats reconstruction in large-scale networks based on alert and log correlation. J Inf Secur Appl 2022;71:103373. http://dx.doi.org/10.1016/j.jisa.2022.103373.
[71] CAPEC - Common Attack Pattern Enumeration and Classification (CAPEC), (2007-2024) URL https://capec.mitre.org/ (Accessed on 6 May 2024).
[72] CWE - Common weakness enumeration, (2006-2024). URL https://cwe.mitre.org/ (Accessed on 6 May 2024).
[73] Sönmez FÖ, Hankin C, Malacaria P. Attack dynamics: An automatic attack graph generation framework based on system topology, CAPEC, CWE, and CVE databases. Comput Secur 2022;123:102938. http://dx.doi.org/10.1016/j.cose.2022.102938.
[74] Mapping EDR to att&cks. 2024, URL https://www.kaspersky.com/enterprise-security/mitre/edr-mapping (Accessed on 6 May 2024).
[75] Symantec Corporation. Symantec endpoint detection and response. 2019, URL https://docs.broadcom.com/doc/endpoint-detection-and-response-atp-endpoint-en (Accessed on 6 May 2024).
[76] Cisco Systems, Inc. Cisco secure endpoint. 2024, URL https://www.cisco.com/site/us/en/products/security/endpoint-security/secure-endpoint/index.html, (Accessed on 6 May 2024).
[77] Sadlek L, Husák M, Čeleda P. Hierarchical modeling of cyber assets in kill chain attack graphs. In: 20th international conference on network and service management. CNSM, 2024, URL https://opendl.ifip-tc6.org/db/conf/cnsm/cnsm2024/1571043250.pdf.
[78] Sadlek L, Yamin MM, Čeleda P, Katt B. Severity-based triage of cybersecurity incidents using kill chain attack graphs: Supplementary materials. 2024, http://dx.doi.org/10.5281/zenodo.14547668, Zenodo.

[79] Hernan S, Lambert S, Ostwald T, Shostack A. Threat modeling – uncover security design flaws using the stride approach. MSDN Mag 2006. URL https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach.

[80] The MITRE Corporation. MITRE D3FEND. 2022, URL https://d3fend.mitre.org/ (Accessed: 23 Aug 2024).

[81] Digital artifact ontology. 2023, URL https://d3fend.mitre.org/dao/ (Accessed: 23 Aug 2024).

[82] Gadyatskaya O, Papuc D. ChatGPT knows your attacks: Synthesizing attack trees using LLMs. In: Anutariya C, Bonsangue MM, editors. Data science and artificial intelligence. Singapore: Springer Nature Singapore; 2023, p. 245–60. http://dx.doi.org/10.1007/978-981-99-7969-1_18.

[83] Attack flow v2.2.3. 2022, URL https://center-for-threat-informed-defense.github.io/attack-flow/(Accessed: 25 Jun 2024).

[84] Cleghorn L. Network defense methodology: A comparison of defense in depth and defense in breadth. J Inf Secur 2013;4(3):144–9. http://dx.doi.org/10.4236/jis.2013.43017.

[85] Sadlek L, Čeleda P, Tovarňák D. Supplementary materials: Identification of attack paths using kill chain and attack graphs. 2022, http://dx.doi.org/10.5281/zenodo.6367986, Accessed on 6 May 2024.

[86] Ou X, Boyer WF, McQueen MA. A scalable approach to attack graph generation. In: Proceedings of the 13th ACM conference on computer and communications security. CCS '06, New York, NY, USA: Association for Computing Machinery; 2006, p. 336–45. http://dx.doi.org/10.1145/1180405.1180446.