SN Computer Science (2022) 3: 347
https://doi.org/10.1007/s42979-022-01209-7

ORIGINAL RESEARCH

Formalizing and Integrating User Knowledge into Security Analytics

Fabian Böhm[1] · **Manfred Vielberth**[1] · **Günther Pernul**[1]

Received: 30 September 2021 / Accepted: 15 May 2022 / Published online: 24 June 2022

Abstract
The Internet-of-Things and ubiquitous cyber-physical systems increase the attack surface for cyber-physical attacks. They exploit technical vulnerabilities and human weaknesses to wreak havoc on organizations' information systems, physical machines, or even humans. Taking a stand against these multi-dimensional attacks requires automated measures to be combined with people as their knowledge has proven critical for security analytics. However, there is no uniform understanding of information security knowledge and its integration into security analytics activities. With this work, we structure and formalize the crucial notions of knowledge that we deem essential for holistic security analytics. A corresponding knowledge model is established based on the Incident Detection Lifecycle, which summarizes the security analytics activities. This idea of knowledge-based security analytics highlights a dichotomy in security analytics. Security experts can operate security mechanisms and thus contribute their knowledge. However, security novices often cannot operate security mechanisms and, therefore, cannot make their highly-specialized domain knowledge available for security analytics. This results in several severe knowledge gaps. We present a research prototype that shows how several of these knowledge gaps can be overcome by simplifying the interaction with automated security analytics techniques.

Keywords Security analytics · Domain knowledge · Visual analytics · Security awareness · Security operations

Introduction

Motivation

Although a lot of money and effort is invested into awareness campaigns and professional training, humans within cybersecurity are still widely considered the weakest link of an organization's cyber defenses [1]. However, this simplification in no way does justice to the role of humans in modern security analytics[1] (SA), because their domain knowledge is invaluable for any effective and efficient SA operation [2, 3]. So far, SA approaches have essentially been limited to integrating the knowledge of security experts to decide, for example, whether identified indicators actually represent malicious incidents or just unusual but benign activities. From our point of view, this is a major shortcoming of existing SA approaches, as it is equally important to include the knowledge of non-security domains in SA processes.

This shortcoming becomes evident in the context of the ever-growing Internet-of-Things (IoT), Industry 4.0, and ubiquitous Cyber-Physical Systems (CPS) (e.g., [4]). All of these trends are leading to increased connectivity of a company's internal and external physical assets. Quite apart from the already skyrocketing number of cyberattacks, the attack surface for cyber-physical attacks is significantly increasing due to this trend. The cyber-physical attacks specifically use the connection between information (cyber) systems and physical systems within CPSs or the IoT to cause actual physical harm to machines or people [5]. Detecting and averting, or mitigating, such multidimensional attacks poses a challenge to existing security measures. To achieve

This article is part of the topical collection "Information Systems Security and Privacy" guest edited by Steven Furnell and Paolo Mori.

✉ Fabian Böhm
fabian.boehm@ur.de

Manfred Vielberth
manfred.vielberth@ur.de

Günther Pernul
guenther.pernul@ur.de

[1] Chair of Information Systems, University of Regensburg, Universitätstr. 31, Regensburg 93053, Bavaria, Germany

[1] Since security analytics has not yet been universally defined we interpret this term as a collection of methods for proactively identifying attacks and threats by analyzing and correlating collected data.

comprehensive security, they must monitor all assets of an organization, which are connected in some way to cyberspace. With the progressive implementation of the IoT and Industry 4.0, these systems range from firewalls or individual sensors to cyber-physical systems, such as complete manufacturing lanes. The problem in the context of these CPS is that traditional security measures and systems, such as a Security Information and Event Management (SIEM) system used in a Security Operations Center (SOC), are not able to sufficiently protect the CPS due to a lack of knowledge and capabilities [6, 7].

Security experts can make well-informed decisions in this context to identify incidents in cyberspace. However, they lack crucial knowledge about the physical domain [8]. For this reason, they often cannot effectively decide whether, for example, a turbine used to generate electricity is operating within its specification or may have a problem [9]. However, engineers and appropriately trained staff have that knowledge of physical operations to decide whether or not the turbine is behaving normally. In turn, however, these employees lack the know-how to contribute to effective SA [7].

This imbalance limits an organization's ability to implement holistic SA methods that could reliably detect indicators of both cyber and cyber-physical attacks. For this reason, it is necessary to integrate the knowledge of engineers and the like into security operations, for example, to enable targeted collaboration between the different types of domain experts. Only then can incidents related to physical assets also be effectively detected and prevented [10]. Although initial approaches exist [6] that integrate data from CPS into a SOC and the importance of interdisciplinary communication has been recognized in other domains [11], to the best of our knowledge there are no efforts yet that attempt to integrate the knowledge of engineers and security experts in a common model.

Contribution and Approach

To the best of our knowledge there is no formal definition of knowledge within security analytics. With this work, we make a twofold contribution to address this problem. To establish a unified and fundamental vocabulary for this research domain, we first define different types of knowledge and knowledge conversions we deem relevant to cybersecurity. We then step forward to provide a holistic view of where the different types of knowledge play a role in Security Analytics. To do so, we first introduce the Incident Detection Lifecycle, which summarizes a series of vital activities for SA. The model for knowledge-based SA is built by highlighting the steps within the Incident Detection Lifecycle, where the different forms of knowledge are necessary for comprehensive security measures. This is a valuable contribution, as no security-specific definition of different knowledge aspects exists so far. Their formal definition can also build future research on a well-defined foundation. Our second contribution addresses quite explicitly the lack of integration of domain knowledge as an open issue within knowledge-based SA. Therefore, we present a research prototype that allows experts to integrate their knowledge into active security measures. This prototype aims at facilitating the externalization of knowledge from non-security domain experts on the one hand and at enabling the collaboration of security experts and security novices on the other hand. This is achieved by applying visual programming approaches and appropriate visual abstractions.

The main contributions of this work are structured according to Fig. 1. The remainder of this paper is organized as follows. We formally describe relevant notions of knowledge and conversion processes for Security Analytics in section "Knowledge Within Security Analytics". These formal definitions allow us and any future work to have a well-defined, precise vocabulary. In the next step, this vocabulary is integrated into an Incident Detection Lifecycle for a cohesive picture of what we call knowledge-based SA within section "Knowledge-Based Security Analytics". Besides several knowledge gaps, the resulting model reveals a significant dichotomy in current SA approaches, which is not yet appropriately addressed. Thus, we present a research prototype in section " Research Prototype" showcasing a possible approach to integrate security novice's domain knowledge into an exemplary SA solution, i.e., a signature-based incident detection component. The prototype highlights that the implementation of knowledge-based SA requires innovative approaches but can improve cybersecurity. Finally, section "Conclusion" concludes our work and points out possible directions for future work.

Extensions from Previous Work

This article is an extended version of our work presented at the 7th International Conference on Information Systems Security and Privacy 2021 (ICISSP, February 2021) [12], kindly invited for consideration in this journal. The first extension from our initial work is a largely extended discussion of the formal knowledge definitions and knowledge conversion processes in sections "Knowledge Types" and "Knowledge Conversion", respectively. This allows for more in-depth and better-structured explanations of these concepts as a foundation for the following sections. The most fundamental extensions have been carried out within section "Knowledge-Based Security Analytics". We first dedicate a newly introduced subsection to the Incident Detection Lifecycle (section "Incident Detection Lifecycle"). Section "Knowledge-Based Security Personas" is also an extension to our original work, which provides, based on feedback and new insights, definitions of the different personas within SA

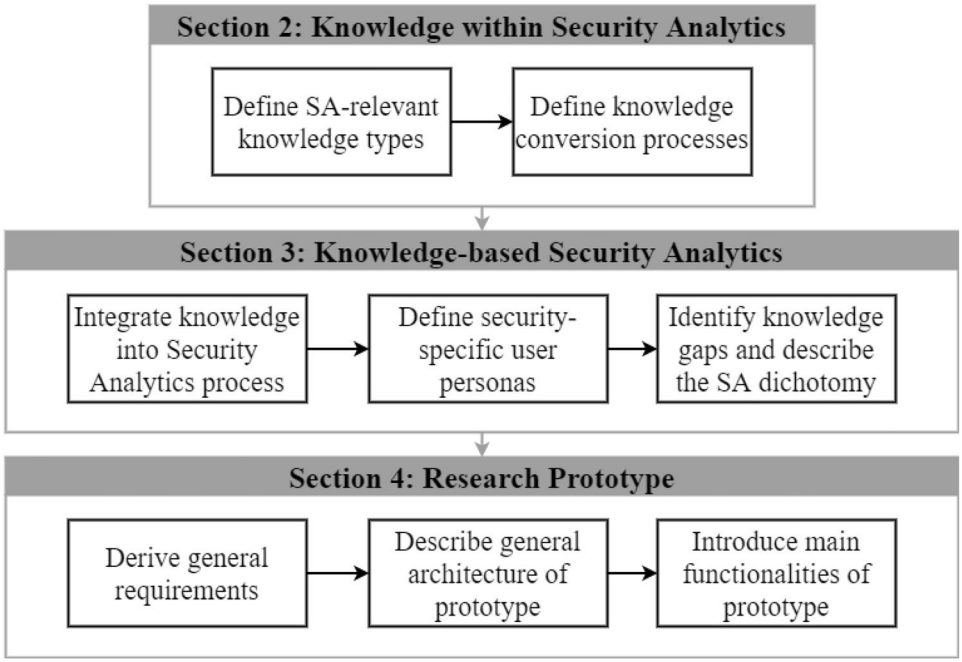

Fig. 1 Schematic of this work's main contribution

concerning their different knowledge sets. All other subsections within section "Knowledge-Based Security Analytics" have been majorly re-worked to provide more detailed information. These subsections now make the model for knowledge-based Security Analytics (section "Knowledge Model") and the inherent dichotomy (sections "Dichotomy of Security Analytics" and "Knowledge Gaps") more comprehensible. Within section "Research Prototype", we added section "Requirements Analysis", which formally derives a set of requirements for a technical solution based on identified knowledge gaps. Thus, it improves the transparency of the aim we pursue with our research prototype. The development of the prototype now follows a straightforward problem-oriented design approach.

Knowledge Within Security Analytics

In this section, we provide a detailed insight into the different knowledge aspects that play a crucial role in the context of current SA operations. For this purpose, we establish a formal understanding of the types of knowledge and the processes for knowledge conversion. While Sallos et al. [13] present the importance of cybersecurity-related knowledge on an abstract and management-oriented level, we aim at the implications of integrating knowledge into security measures in the following sections.

Knowledge Types

Scientific literature describes a variety of different, sometimes even contradictory, definitions of the term "knowledge" and the different sub-aspects related to it. A frequently cited definition that provides a clear starting point for opening up the concept of knowledge is the data-information-knowledge-wisdom (DIKW) hierarchy, which defines "knowledge" as the application of data and derived information to answer "how" questions [14]. However, information systems research often criticized the DIKW hierarchy as unsound and undesirable [15]. A more human-oriented definition by Davenport describes knowledge as a mixture of experience, intuition, values, contextual information, and expert insight [16].

This definition of Davenport appropriately explains the concept of knowledge from a human point of view; however, knowledge is not bound to humans. Instead, corresponding research emphasizes that knowledge can also be captured in documents, memos, and the like [17]. Following this route, it is only logical to conclude that knowledge can also be stored within IT. This knowledge within IT is different from human knowledge, especially if it is also generated by IT through some kind of automatic analysis [18, 19]. Based on this line of thought, it is an established and accepted procedure to distinguish between two basic types of knowledge: explicit knowledge and tacit (or implicit) knowledge [17].

All these aspects clearly show that the term "knowledge" is difficult to define in a generally valid way. Instead, different facets of knowledge must be distinguished and embedded

in the relevant context. This is because even the notions of explicit and tacit knowledge are still too abstract in their basic form to be incorporated into processes of SA. For this reason, we define and formalize below different notions of knowledge that are of central importance in the field of cybersecurity and even more specifically in the field of SA.

Explicit Knowledge

Explicit knowledge K^{e} is mainly referred to as machine-based knowledge. Accordingly, this term denotes knowledge that machines can read, process and store [17]. In the context of SA, we distinguish three types of explicit knowledge in the further course of the work, which can be distinguished from each other by their intended use for SA. The transitions between these different types of explicit knowledge are fluid, i.e., a given machine-readable object can also be assigned to a different expression depending on the current context and use case. Equation (1) defines explicit knowledge formally as a union of the three sub-aspects defined in the following paragraphs.

$$K^{\mathrm{e}} = K^{\mathrm{e}}_{\mathrm{m}} \cup K^{\mathrm{e}}_{\mathrm{s}} \cup K^{\mathrm{e}}_{\mathrm{i}} \tag{1}$$

Models ($K^{\mathrm{e}}_{\mathrm{m}}$): Models for machine learning approaches, neural networks, and the like are primarily used for anomaly-based detection mechanisms. This knowledge allows a machine to detect outliers and evaluate them to some extent as to whether they indicate malicious or undesirable behavior.

Signatures & Rules ($K^{\mathrm{e}}_{\mathrm{s}}$): like models for machine learning approaches, signatures and rules are also to be valued as explicit knowledge, especially in signature-based security analytics methods. They are the basis for more traditional SA approaches such as SIEM systems and their correlation engines for detecting indicators of compromise (IoC).

Threat Intelligence & Forensic Evidence ($K^{\mathrm{e}}_{\mathrm{s}}$): Threat Intelligence and forensic evidence describe the results of primarily manual, in-depth analysis of suspected or actual incidents and include extensive information on the attackers' modus operandi, identifiable traces, suspect groups or individual perpetrators, and many other details. Because of their level of detail, Threat Intelligence and Forensic Reports allow answering "how" questions.

Implicit Knowledge

After contextualizing explicit knowledge in SA, we turn to so-called tacit knowledge in the following paragraphs. This kind of knowledge can only be possessed by humans and is very specific to each individual [20]. Although "tacit knowledge" would be a more commonly used term, we will use "implicit knowledge" K^{i} in this paper to clarify the distinction from the explicit knowledge of a machine.

Humans improve their K^{i} by combining new insights with existing knowledge. The existing knowledge itself can in turn be divided into, on the one hand, domain knowledge and, on the other hand, operational knowledge [21]. However, in the context of SA, we consider this differentiation too vague. To describe the problem at hand concisely, a more fine-granular and contextualized view on K^{i} is necessary. In the domain of SA, we also consider another new type of tacit knowledge to be highly relevant: situational knowledge. As for explicit knowledge, we also define implicit knowledge as a union of its three main facets (c.f. Eq. 2). We go into more detail about these three aspects of tacit knowledge in the following paragraphs.

$$k^{\mathrm{i}} = K^{\mathrm{i}}_{\mathrm{d}} \cup K^{\mathrm{i}}_{\mathrm{s}} \cup K^{\mathrm{i}}_{\mathrm{o}} \tag{2}$$

Domain Knowledge $\left(K^{\mathrm{i}}_{\mathrm{d}}\right)$: Generally speaking, domain knowledge describes what people know about a particular context or on a specific topic (the "domain") [2, 7]. For SA, we define $K^{\mathrm{i}}_{\mathrm{d}}$ in Eq. 3 in a more detailed way as a combination of two disjoint subdomains $K^{\mathrm{i}}_{\mathrm{d(sec)}}$ and $K^{\mathrm{i}}_{\mathrm{d(nonSec)}}$. $K^{\mathrm{i}}_{\mathrm{d(sec)}}$ comprises security-related domain knowledge, which is mainly part of the tacit knowledge of security experts. The components of $K^{\mathrm{i}}_{\mathrm{d(sec)}}$ are all safety and security aspects considered from a cybersecurity perspective. For example, this includes knowledge about firewall rules in use, the ability to identify suspicious network connections or unauthorized access to classified information. This facet of domain knowledge is to some extent already considered in several SA means [22]. In contrast, there is a lack of integration of $K^{\mathrm{i}}_{\mathrm{d(nonSec)}}$. Under this second aspect of general domain knowledge, we summarize non-security domain knowledge. This type includes domains, such as manufacturing or engineering. The knowledge from these domains is of high importance to detect incidents on cyber-physical systems [6]. An example of domain knowledge not directly related to security is the expected Rounds per Minute (RPM) of a power turbine or the maximum temperature for a blast furnace. However, in the context of SA, this knowledge is necessary to assess the CPS's security posture cohesively. For SA, both components of domain knowledge are necessary to build and operate comprehensive security operations. Especially in light of the challenges associated with CPS and the rise of cyber-physical attacks, the integration of $K^{\mathrm{i}}_{\mathrm{d(nonSec)}}$, in particular, is one of the biggest challenges currently faced by SA research.

$$K^{\mathrm{i}}_{\mathrm{d}} = K^{\mathrm{i}}_{\mathrm{d(sec)}} \cup K^{\mathrm{i}}_{\mathrm{d(nonSec)}} \tag{3}$$

Situational Knowledge $\left(K^{\mathrm{i}}_{\mathrm{s}}\right)$: Situational knowledge is a new type of knowledge previously not acknowledged, which we

consider crucial in the SA environment. In SA, this type mainly encompasses the concept of situational awareness, which also plays a vital role in cybersecurity in recent years, especially in the form of Cyber Situational Awareness [23, 24]. K^{i}_{s} describes the ability of any employee of an organization to perceive unusual events or suspicious behavior. The relevant events range from receiving suspicious mail, which represents a possible phishing attempt, to identifying a private storage medium connected to a corporate device. With the appropriate situational security knowledge, which has been imparted, for example, through security awareness training or campaigns [25], employees can evaluate the e-mail or the storage medium from a security perspective and deduce that these events could pose a threat to the company. However, specific domain knowledge K^{i}_{d} about the SA of the enterprise is not required to make these inferences.

Operational Knowledge $\left(K^{i}_{o}\right)$: Operational knowledge in the context of SA refers to the ability of a human to operate specific systems. Specifically, employees with SA-related operational knowledge can adequately operate a company's security systems. This ability can relate to a wide variety of systems. For example, employees may have the experience to define correlation rules for a SIEM, fine-tune models for anomaly- or behavior-based SA approaches, or create new threat intelligence. It is important to note here that K^{i}_{o} does not refer to expertise, such as the syntax of the threat intelligence format used, but rather to the ability to deal with the corresponding IT system.

These three different subsets of tacit knowledge are necessary to detect and resolve both cyber and cyber-physical attacks as completely as possible. K^{i}_{d} and K^{i}_{s} would, in a perfect world, need to be comprehensively integrated into an organization's SA systems. They are the pre-requisite to cohesive security operations, especially in the context of CPS and IoT. However, operational knowledge K^{i}_{o} represents the barrier to entry for this integration. Only with the necessary K^{i}_{o} can employees, for example, define an appropriate SIEM correlation rule based on their K^{i}_{d}.

Knowledge Conversion

The different knowledge types can be converted into each other. Nonaka and Takeuchi define the knowledge conversions between explicit and tacit knowledge in terms of four different knowledge conversion processes [17]. Various research directions have picked up upon this formalization to formally describe the exchange and interaction between humans and machines. Especially research in the field of information visualization and human–machine interaction work frequently and intensively with these concepts [22, 26, 27]. In SA, corresponding knowledge exchange is also desirable in the underlying approaches, since effective security operations require both automated discovery processes (involving explicit knowledge) and the expertise of different human experts (and their tacit knowledge). To provide the necessary foundation and common vocabulary regarding knowledge conversion in SA after defining the aspects of knowledge, we formalize the four key knowledge conversion processes for SA in the following paragraphs.

Internalization (int): internalization describes the process of making explicit knowledge available to users, who can then perceive this knowledge using the K^{i}_{o} available to them and convert it into $K^{i}_{d(sec)}$ or K^{i}_{s} (Eq. 4). How efficient this process is and how significant the increase in implicit knowledge is, depends strongly on the respective user's level of K^{i}_{o}. We have implied this dependence in the formal definition in Equation 4 by defining operational knowledge as a catalyst for the int conversion process. This notation is adopted from the formal descriptions of chemical reactions. Effective internalization int of K^{e} is supported primarily by any kind of visual representation of the K^{e}. For security-related domain knowledge, this includes examples like visualizing the raw data that led to the triggering of a SIEM rule, the rule itself, and the components of the data that were conducive to the decision.

$$\text{int} : \left(K^{e} \overset{K^{i}_{o}}{\mapsto} K^{i}_{d(sec)} \cap K^{i}_{s} \right) \tag{4}$$

Externalization (ext): When tacit knowledge, especially K^{i}_{d} or K^{i}_{s}, is transferred into a form that can be processed by computers, we refer to this as the process of externalization (Eq. 5). Externalized tacit knowledge can thus be read, persisted, and eventually processed by computers. A variety of examples for externalization can be found in the context of modern security analytics. For example, this process includes the direct adaptation of model parameters (i.e., K^{e}_{m}) and the formulation of rules for signature-based analysis (i.e., K^{e}_{s}). Structuring and formalizing indicators, incidents, and corresponding evidence into CTI (i.e., K^{e}_{i}) also represents a form of externalization. Here, direct access to explicit knowledge and possibilities for active processing of the same are of primary importance. Thus, the corresponding operational knowledge K^{i}_{o} is again a fundamental prerequisite for enabling and performing externalization. Only if the human being can operate a system (e.g., SIEM system with the corresponding correlation rules), the possibility to externalize implicit knowledge can be retained. The process described here for translating tacit to explicit knowledge is also necessary for avoiding the loss of any K^{i} due to, for example, the retirement of a security analyst from the company. If there is no possibility to keep the knowledge of this security analyst in the company, this poses a risk for the company [28].

$$\text{ext} : \left(K^{i}_{d} \cap K^{i}_{s} \overset{K^{i}_{o}}{\mapsto} K^{e} \right) \tag{5}$$

Combination (comb): The conversion process of combination describes the exchange of knowledge from two or more explicit knowledge bases (Eq. 6). At the same time, it can also mean the exchange of knowledge between corresponding K^{e}. Concerning the explicit knowledge types defined in section "Explicit Knowledge", *comb* can describe both the exchange of knowledge within a constant knowledge type but also the transfer of knowledge from one type to another. An example of the first process is the exchange of cyber threat intelligence (CTI) and forensic evidence between different actors $\left(K_{i}^{e} \mapsto K_{i}^{e}\right)$. The second process may be, for example, using CTI to define new or adapt existing rules for signature-based incident detection $(K_{i}^{e} \mapsto K_{s}^{e})$.

$$\text{comb} : K^{e} \mapsto K^{e} \tag{6}$$

Collaboration (coll): In the context of collaboration, multiple individuals work together and combine their K^{i} (Eq. 7). Less formally, this knowledge conversion specifies that people can learn from each other (i.e., increase their K^{i}) by collaborating. This process is difficult to capture and formally define, because it is purely implicit without any direct indication that it is happening. Even a simple conversation between two people can correspond to a knowledge conversion. However, we interpret collaboration in the context of SA as a process that is supported by technology. Accordingly, operational knowledge of collaborators is again required to enable collaboration, as also indicated in Eq. (7) by K_{o}^{i} as the catalyst of collaboration. Collaborators can thus work together, for example, in correlating various indicators of compromise to determine which indicators genuinely represent a threat. To do this, they could use an appropriate analysis tool designed for just such a purpose. On the one hand, the tool support enables remote collaboration among the employees, but at the same time, they need the operational knowledge to be able to operate this tool. Collaboration supported in this technological way enables users to share knowledge and learn from each other. With respect to SA and the need to involve $K_{d(sec)}^{i}$ and $K_{d(nonSec)}^{i}$, appropriate knowledge sharing is vital between collaborators to enable the most comprehensive SA possible. In addition, for example, tool-based training or workshops can be defined as a type of collaboration. These workshops often impart domain knowledge to improve the situational knowledge of other collaborators $\left(K_{d}^{i} \mapsto K_{s}^{i}\right)$.

$$\text{coll} : K^{i} \overset{K_{o}^{i}}{\mapsto} K^{i} \tag{7}$$

Knowledge-Based Security Analytics

After a basic understanding of the formal knowledge types and the processes describing the conversion among these knowledge types is established in section "Knowledge Within Security Analytics", the following section is dedicated to embedding these concepts into the core activities of security analytics. In this context, we interpret the detection of security incidents, i.e., attacks on an organization's assets, to be the the essential task of SA [29]. A cohesive approach to implementing this task requires comprehensive data collection combined with powerful analytical capabilities and the integration of any available knowledge base. We summarize these activities within the *Incident Detection Lifecycle* (IDL) as a general process containing the core activities of SA. Thus, we introduce our model for knowledge-based SA based on the IDL and the critical role that knowledge plays in its context. Based on this, we identify different personas of users that play a role in knowledge-based SA. Finally, we explain the central problem faced by SA in the context of current developments, such as the Internet of Things and Industry 4.0, which we refer to as the "Dichotomy of Security Analytics".

Incident Detection Lifecycle

The starting point for our model of knowledge-based security analytics is the incident detection process defined by Menges and Pernul [30]. This process describes four basic steps involved in incident detection: *Data*, *Observables*, *Indicators*, and *Incidents*. *Data* of a system under consideration are collected and normalized, resulting in so-called *Observables*. The authors refer to detected anomalies in these observables as Indicators of Compromise or just *Indicators*. Only the combination of several indicators finally confirms a recognized textitIncident. While this simple model describes the core activities for detecting security incidents, it neglects two central aspects of modern security analytics. First, an incident detection is usually followed by a post-incident analysis to extract and secure forensic evidence. Second, the subsequent analysis of an incident can also serve to generate threat intelligence, which can again be used to detect indicators or specific incidents. Incident detection is thus an iterative process in which the output (threat intelligence) can be used as an input in further detection runs. For this reason, we are extending the original Incident Detection Process to an Incident Detection Lifecycle, which more appropriately reflects the processes within modern security analytics.

Figure 2 represents this adapted and extended lifecycle. The boxes highlighted in gray represent the central results of the activities. The SA activities themselves are

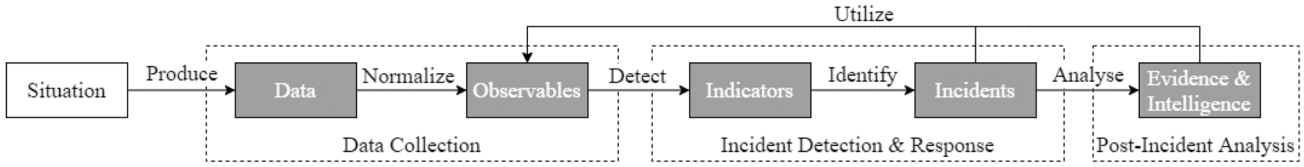

Fig. 2 Incident detection lifecycle

annotated at the edges of the model. The starting point of the Incident Detection Lifecycle is some real event within an organization—that is, something that "happens"—which can be physical or digital. Examples of such a *situation* are the authentication of a user at an IT system or the use of a private USB stick at company computers. We refer to these events as *situations* in the further course. The Incident Detection Lifecycle is divided into three overarching phases, which are executed to detect, resolve, and understand incidents. Overall, the Incident Detection Lifycycle provides a more detailed view on the Detect and Respond phases of the established NIST Cybersecurity Framework [31].

The first of these phases is the *Data Collection*. Each *situation* produces raw data which could be relevant for the detection of possible attacks. These *data* are normalized (and sometimes standardized) in the first phase of the lifecycle, producing so-called *observables*. Observables can thus be understood as normalized representations of the raw data available about the situation. They are not yet attributed and thus have no significance for why something happened or who might be responsible for it. Observables only serve as input for the second phase of the Incident Detection Lifecycle.

This second phase of the lifecycle can be summarized under the terms *Incident Detection & Response*. This phase aims to detect actual incidents, capture the impact, and contain the incident as quickly as possible. The first step is the detection of *indicators*, which are often also referred to as Indicators of Compromise (IoC). These indicate potentially suspicious activities and behaviors within the observables. However, IoCs can also indicate unusual but not malicious behavior. For this reason, a further step is necessary to identify actual *incidents* from detected indicators. For this purpose, it is necessary to correlate indicators with each other and possibly to include additional data or observables in the analysis process. However, if an incident is identified, direct measures for defense and containment must be initiated in this lifecycle phase.

After the initiation and implementation of countermeasures and containment actions, the third phase of the Incident Detection Lifecycle, the *Post-Incident Analysis*, is carried out. In this phase, careful and intensive analyses of an incident produce further vital artifacts. On the one hand, *evidence* which can be used in possible judicial proceedings is collected in this step through forensic analysis. On the other hand, *threat intelligence* is generated through the attribution of the identified incident. Since the gained intelligence can also be crucial to detect new indicators or identify similar incidents, it feeds into the previous phases, creating an iterative lifecycle.

Knowledge Model

The Incident Detection Lifecycle can now be extended to a model for knowledge-based SA in the next step. In the course of this extension, the knowledge terms and conversion processes introduced in section "Knowledge Within Security Analytics" are integrated into the lifecycle to obtain a comprehensive picture of the stages in the lifecycle at which knowledge and knowledge exchange play a central role. The extended model is shown in Fig. 3. In the following paragraphs, we will go through this knowledge model in detail to highlight the significant adjustments made compared to the original Incident Detection Lifecycle.

To recognize indicators in a considerable amount of observables and, above all, to derive correct indicators is an enormously challenging task. Due to the sheer amount of observables that need to be monitored, this task is primarily automated in modern SA systems [29]. The corresponding processes in the *Incident Detection & Response* phase of the lifecycle use explicit knowledge K^{e} in the form of signatures or rules (K^{e}_{s}) for signature-based detection, but also models (K^{e}_{m}) for behavior-based procedures. Thus, explicit knowledge plays a central role, especially for incident detection. Nevertheless, a pure focus of incident detection on K^{e} is not purposeful and can even be associated with direct limitations. First of all, with the use of K^{e}_{s} only indicators and incidents that were known apriori and whose signatures were integrated into the system, can be detected. Behavior-based methods are better at classifying unknown indicators but often tend to generate a large number of false positives. By incorporating human domain experts, these two fundamental problems can be eliminated or at least mitigated to some extent. On the one hand, experts can analyze parts of the available observables to discover new, previously unknown indicators. On the other hand, humans can use their domain

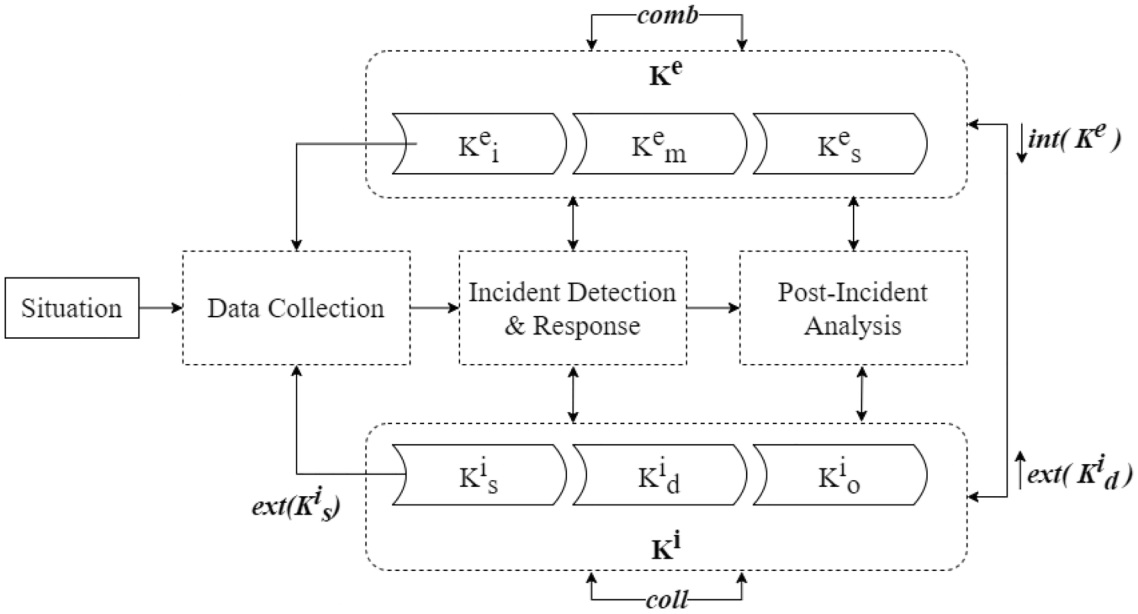

Fig. 3 Incident detection lifecycle extended with knowledge types and conversions

knowledge K^{i}_{d} to decide whether an indicator ultimately describes a destructive action or not. For this reason, integrating K^{i}_{d} at this stage is highly beneficial and can even be considered inevitable for any approach of effective SA.

Within this phase of the Incident Detection Lifecycle, the next step is to identify incidents within the previously recognized indicators. In this step, the involvement of K^{i} is even more critical than for the detection of indicators. The main reason for this is that utilizing K_{e} in this step can only detect previously known attacks for which the corresponding K^{e}_{s} has already been defined. For new, unknown attack procedures, K^{e}_{s} cannot contribute, and also, K^{e}_{m} can hardly detect more than indicators that point to potentially malicious activity. In this context of actual attack detection, K^{e} cannot capture an incident to its full extent. Again, the involvement of human domain experts is necessary. Only this human component with K^{i}_{d} can analyze various indicators in their context, correlate them, and ultimately distinguish between malicious and regular activity. In summary, K^{e} in its various forms can contribute significantly to detecting indicators in the observables and thus reduce information overload. However, a final interpretation and classification of the indicators and the associated indicating of concrete incidents is not effectively possible in the vast majority of cases without direct integration of K^{i}_{d}.

While automatic analysis using K^{e} plays a major role in the first two phases of the Incident Detection Lifecycle, this focus shifts in the final phase, the *Post-Incident Analysis*. In this step, almost exclusively manual work steps take place in the context of forensic investigations and the attribution of incidents. Thus, the influence of K^{e} is rather low compared to K^{i} and the integration of K^{i} into automated workflows is stronger.

In addition to the inclusion of both K^{e} and K^{i} in the Incident Detection Lifecycle, we have indicated several other knowledge conversion processes in Fig. 3. We identify all these additional processes as relevant and necessary for a comprehensive and effective implementation of SA, which covers both the cyber domain and the cyber-physical domain. Some of the processes plotted have already been presented in detail in section "Knowledge Conversion": int(K^{e}) (Eq. 4), comb (Eq. 6), and coll (Eq. 7). For this reason, we focus on the two remaining processes: ext(K^{i}_{s}) and ext(K^{i}_{d}). They are each an instance of ext, but require a closer, contextualized look.

Externalization of situational knowledge K^{i}_{s} (ext(K^{i}_{s})) fundamentally allows employees to feed events (i.e., situations) they have observed or experienced into the SA system as observables. This allows the semantic information transformed from K^{i}_{s} into observables by ext(K^{i}_{s}) to be used in the further steps of the Incident Detection Lifecycle. If this possibility is exploited efficiently, it significantly expands the availability of observables for SA, because many aspects of targeted attacks are not detected in automatically collected data. Examples are social engineering attacks or direct physical access attempts. Information about these and a multitude of other attack vectors cannot be collected through automated data collection mechanisms. With the ability to externalize ext(K^{i}_{s}), virtually every employee turns into an extremely valuable source of observables for incident detection when, for example, the employee reports a phone call

attempting to discover critical access privileges. A unique feature of this conversion process is that it does not build on domain knowledge K^{i}_{d}, but a more general knowledge that comes primarily from situational awareness K^{i}_{s}. The externalization of situational knowledge is particularly relevant, because it is the only way to fully capture possible attack vectors involving the physical aspects of modern attacks.

It is also necessary, to take a closer look at the process $\text{ext}\left(K^{i}_{d}\right)$. This activity basically comprises two processes: $\text{ext}\left(K^{i}_{d(sec)}\right)$ and $\text{ext}\left(K^{i}_{d(nonSec)}\right)$. It thus describes the interaction of humans with the analysis processes sustained by K^{e}. The fundamental goal of this interaction is to integrate K^{i}_{d} into security analytics, thus making human domain knowledge available to improve the overall incident detection lifecycle. In the era of cyber-physical systems, domain knowledge, specifically $\text{ext}\left(K^{i}_{d(nonSec)}\right)$, is widely distributed across enterprises. At the same time, however, the entirety of domain knowledge is necessary for comprehensive and effective security analytics. For this reason, the $\text{ext}\left(K^{i}_{d}\right)$ process is critical as it is the only way to translate human knowledge into SIEM correlation rules, attack signatures, or improved behavior models for the organization's assets.

Another aspect that stands out in Fig. 3 is the exclusion of the *Utilize* loop, which illustrates the iterative nature of the Incident Detection Lifecycle in Fig. 2. However, a closer look at Fig. 3 reveals that this process step is by no means missing but has only been made more precise by integrating K^{e} and the corresponding conversion processes into the representation. Through the bi-directional connections between the lifecycle phases *Incident Detection & Response* and *Post-Incident Analysis* as well as K^{e}, our knowledge model makes clear that in these phases, K^{e} can be used and at the same time also generated. The K^{e} generated in these phases can be defined more precisely as the K^{e}_{i} described in previous sections. K^{e}_{i} serves as input to the *Data collection* phase, thus preserving the iterative nature of the life cycle.

Knowledge-Based Security Personas

Based on the various security-related knowledge types, different groups of users can be distinguished. As shown in Eq. (8) the knowledge of users can be seen in this context as different instances of K^{i}.

$$k^{i}_{d(nonSec)}, k^{i}_{d(sec)} \in K^{i}_{d}, k_{s} \in K^{i}_{s}, k^{i}_{o} \in K^{i}_{o} \tag{8}$$

In an organizational context, employees can essentially be assigned to two roles from an SA perspective, which can be referred to as security personas: security novices S_{n} and security experts S_{e}.

- Security novices: In general, a novice is a user without profound knowledge and experience within a specific domain. In our case, S_{n} are employees without deeper knowledge in security. However, in practice, a clear differentiation is not always easy, since almost everyone has a basic sense of security. It is easier to make a distinction by taking an employee's areas of activity into account. Security novices can be defined as persons who do not deal with security in their daily activities, or only to a very limited extent (like for example gained from participating in awareness programs). From a knowledge perspective, S_{n} have domain knowledge in a domain other than security: $k^{i}_{d(nonSec)}$. This domain knowledge can be very individually pronounced from user to user. An example would be engineering knowledge, if a user is responsible for maintaining a turbine and thus knows precisely how it works. From a security perspective, situational knowledge k^{i}_{s} of S_{n} is particularly relevant, as it enables them to judge a situation in combination with their unique $k^{i}_{d(nonSec)}$. Thus, they can contribute to the Incident Detection Lifeccycle by observing and reporting possible attack vectors. As shown in Fig. 4a, however, S_{n} have very few connection points with the Incident Detection Lifecycle, which is mainly due to the lack of k^{i}_{o}. $\text{ext}(K^{i}_{s})$ is possible if the respective system is sufficiently simple to use, thus, this process is present in the figure but grayed out:

$$S_{n} = \left\{k^{i}_{d(nonSec)}, k^{i}_{s}\right\} \tag{9}$$

- Security experts S_{e}, in contrast, are employees with in-depth security-related domain knowledge $k^{i}_{d(sec)}$. This usually results from the significant involvement with security issues in their day-to-day business (for example as an employee within a Security Operations Center). Their $k^{i}_{d(sec)}$ in combination with k^{i}_{s} enables them to identify security incidents at a high level of detail and to realistically assess its extent and severity. In addition, they have the necessary operational knowledge k^{i}_{o} to operate security systems (such as SIEM systems) that are used for automated analyses within the Incident Detection Lifecycle. This results in a S_{e} being the main gateway to the Incident Detection Lifecycle (see Fig. 4b). Both $\text{ext}(K^{i}_{s})$ and $\text{ext}(K^{i}_{d(sec)})$ are possible, since the expert has the necessary k^{i}_{o} to comprehensively operate the systems involved.

$$S_{e} = \left\{k^{i}_{d(sec)}, k^{i}_{s}, k^{i}_{o}\right\} \tag{10}$$

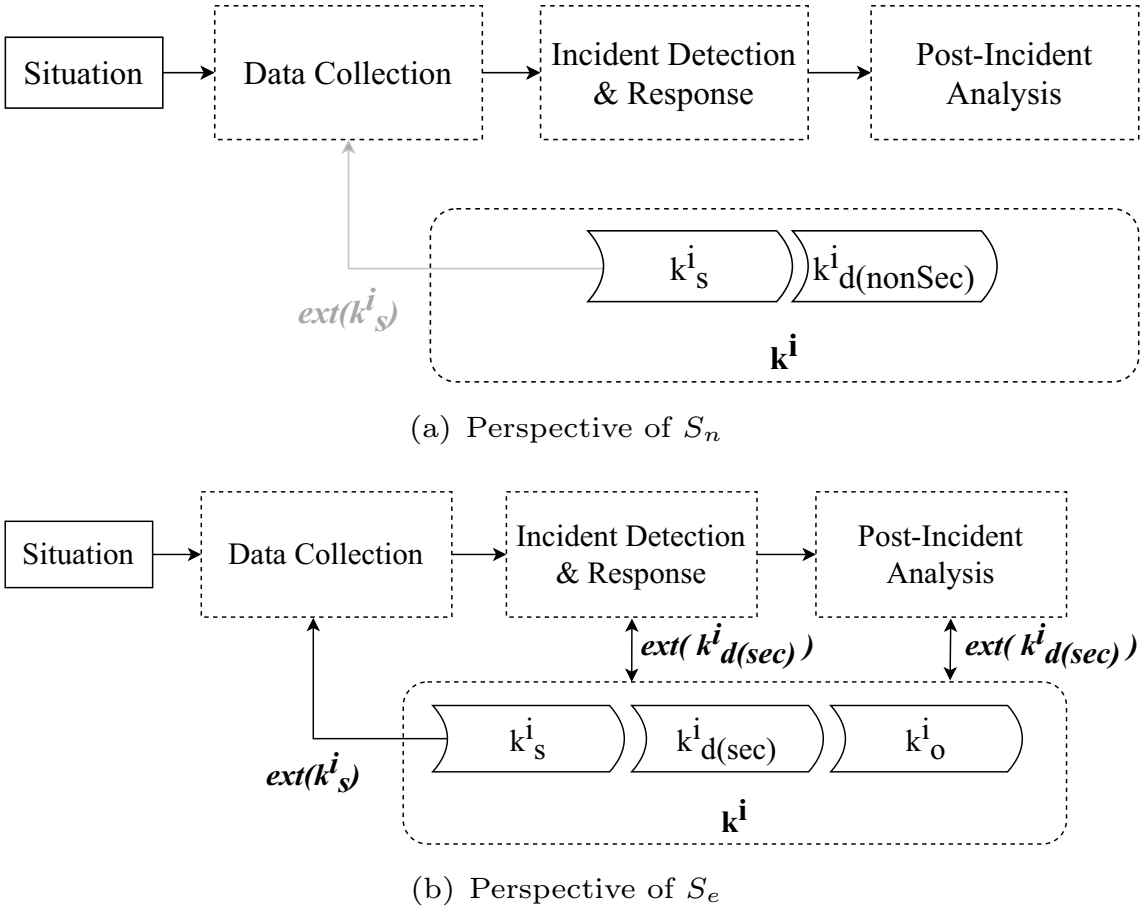

Fig. 4 Knowledge Model from the perspective of the two personas

Dichotomy of Security Analytics

The previous breakdown of the two security personas already highlights the different types of knowledge that are divided between the two personas. It is particularly noticeable here that neither of the two combines all knowledge and thus the knowledge required for the Incident Detection Lifecycle in one person. This circumstance indicates what we call the dichotomy of SA. When comparing the knowledge sets of S_n and S_e, it is noticeable that the differences can essentially be broken down to two knowledge types: Domain knowledge k^i_d differs ($k^i_{d(sec)}$ vs. $k^i_{d(nonSec)}$) and S_n has no or very little operational knowledge k^i_o.

As already defined in Eq. (5) the externalization of implicit knowledge is the intersection of k^i_d and k^i_s. However, when considering cohesive incident detection in the era of cyber-physical attacks, the necessary domain knowledge k^i_d is distributed between S_n and S_e in the form of $k^i_{d(sec)}$ and $k^i_{d(nonSec)}$. Cyber-physical incidents are only detectable if knowledge about security incidents in general ($k^i_{d(sec)}$) and knowledge about the physical aspects in particular (

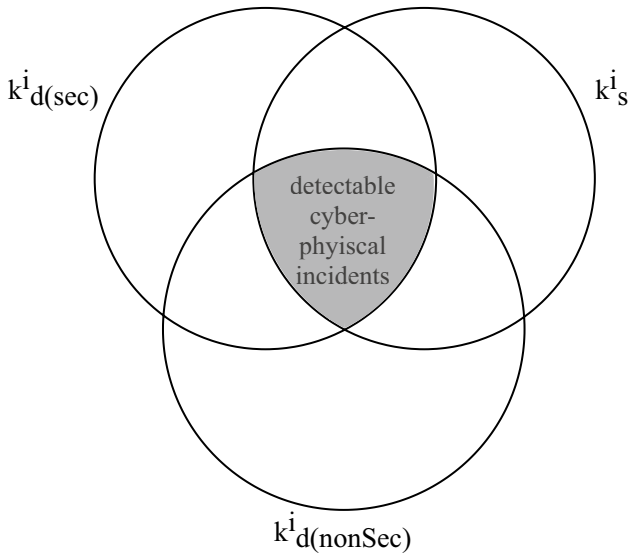

Fig. 5 Required knowledge types for incident detection

$k^{i}_{d(nonSec)}$) are combined. In addition, situational knowledge k^{i}_{s} is necessary for the incident to be recognized in the first place. Therefore, only incidents for which all three types of knowledge are combined can be detected. Figure 5 shows this relationship as an intersection. All incidents that do not reside on the intersection cannot be detected by humans, which is why these areas potentially constitute a blind spot in the Incident Detection Lifecycle and thus have to be minimized.

Operational knowledge k^{i}_{o} takes on a special role in the context of the dichotomy, since it is highly dependent on the security systems in use. Since these are usually expert systems, it is assumed that only security experts have the necessary knowledge to operate them properly. However, these systems should aim to be so easy to use that they require only little operational knowledge to empower S_n to contribute to security operations. Therefore, operational knowledge ideally is kept to a minimum in practice, in contrast to the other types of knowledge.

Knowledge Gaps

The dichotomy in SA creates some knowledge gaps, some of which have already been alluded to in section "Dichotomy of Security Analytics". Essentially, three knowledge gaps limit the Incident Detection Lifecycle or prevent security incidents from being detected. In the following, these gaps are described in detail to highlight a path to close them:

1. $\text{ext}(K^{i}_{S})$: The first gap that can be identified is the lack of possibilities to externalize K^{i}_{s}. The main difficulty here is how S_n can be incorporated appropriately or to create the means to do so. For example, if an employee notices a security incident, they need to be able to contribute their observations to the Data Collection phase of the Incident Detection Lifecycle. Initial approaches to this already exist in the form of the human-as-a-security-sensor paradigm [32, 33]. However, further research is needed in this direction to solve this problem in an applicable way.
2. $K^{i}_{d(nonSec)}$: The next gap stems from the aforementioned k^{i}_{o}, which is not held by S_n in necessary amounts. Therefore, it must be ensured that the required k^{i}_{o} is reduced so that people without expert knowledge can operate security mechanisms. For example, it should be possible to involve engineers who know precisely how a turbine works and what security incidents can look like in the Incident Detection Lifecycle. However, it is unlikely that this problem will be solved entirely. For example, even with a great deal of effort, it will hardly be possible for engineers to create correlation rules for SIEM systems, as these are relatively complex by nature. Therefore, these systems must be simplified to the extent that S_n can at least contribute their knowledge in a simplified manner to contribute to the definition of meaningful rules.
3. coll: Collaboration between the actors, especially between S_n and S_e, within the Incident Detection Lifecycle, is vital, because, as elaborated in section "Conclusion", knowledge is not concentrated on individual persons but is distributed among several personas. Collaboration between the various personas can help create a central knowledge base in the Incident Detection Lifecycle in which as much relevant information as possible is brought together. The knowledge gaps mentioned in (1) $\left(\text{ext}(K^{i}_{S})\right)$ and (2) $\left(K^{i}_{d(nonSec)}\right)$ can help to enable or at least simplify collaboration. Collaboration has not yet been considered much in SA research, although it plays a significant role within the Incident Detection Lifecycle.

Research Prototype

The gaps described in section "Knowledge Gaps" are not yet addressed explicitly in existing work. For this reason, in the following section of our paper, we present the second part of our contribution: a research prototype for a signature-based incident detection system that supports the two above-mentioned conversion processes. The concept and structure of the prototype are built according to the model of knowledge-based SA (see Fig. 3). To detect indicators and identify incidents from their context, we apply a Complex Event Processing approach, which can be based on an arbitrarily complex pattern hierarchy. This hierarchical approach initially allows the detection of indicators based on observables. Additional and more advanced patterns are then used to identify actual incidents by correlating IoCs. The patterns, i.e., signatures needed for this purpose, correspond to K^{e}_{s} in the context of knowledge-based SA and are made accessible to humans by the prototype.

In the following sections, we first derive general requirements. We then present our prototype's system architecture and detail two essential components that are central to address the knowledge conversion processes. For the sake of clarity, we use the term "event" whenever it is not necessary to distinguish specifically between observable, indicator, or incident.

Requirements Analysis

In the age of CPS and IoT, one of the most pressing obstacles to overcome in the quest for holistic Security Analytics is to minimize the tremendous amount of K^{i}_{o} necessary to implement $\text{ext}(K^{i}_{d(nonSec)})$. This can be achieved by providing cen-

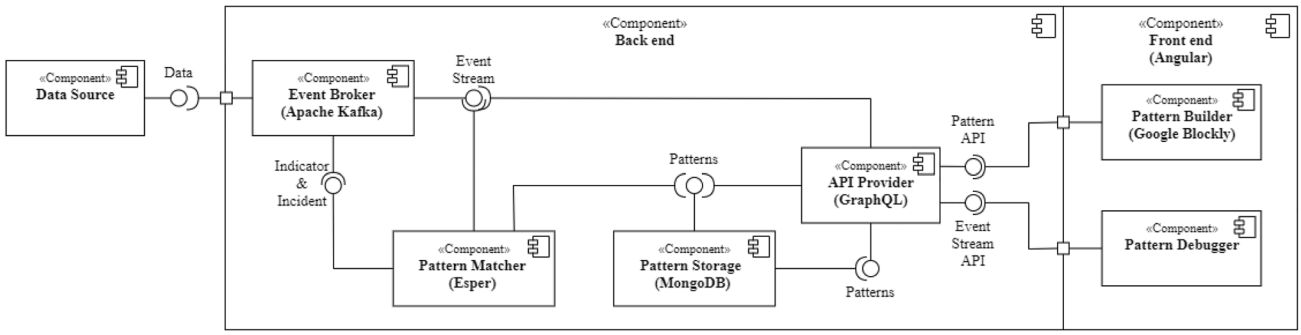

Fig. 6 Component diagram of the architecture for visual collaborative pattern definition [12]

tralized, interactive visual access to the K^{e} underlying the lifecycle. In addition, the next step is to provide better technical support for collaboration, or coll, between people. These two problems are summarized by the following paragraphs in their immediate context:

1. *Reduce the needed* K^{i}_{o} *f*or ext(K^{i}_{d}): K^{i}_{o} is required for all sub-aspects of the int and ext processes. However, since it has no direct impact or purpose for cybersecurity itself, the K^{i}_{o} required to operate security systems should be reduced as much as possible, especially for S_{n}. Besides offering training for S_{n}, this is the only possible way towards better integration of $K^{i}_{d(nonSec)}$. Novices are not skilled in dealing with security solutions, such as a company's SIEM system. Therefore, for the integration of their $K^{i}_{d(nonSec)}$, the entry barrier to these systems (i.e., $K^{i}_{d(nonSec)}$) should be kept as low as possible. Thus, concerning the chosen notation of K^{i}_{o} as a catalyst for knowledge conversion, it is necessary to reduce the "need" for the catalyst as much as possible.
2. *Enable* coll *b*etween S_{e} *a*nd S_{n}: while security experts own knowledge of a variety of possible attack vectors, the knowledge of adapting these attack vectors for a particular context is often within the scope of activity and knowledge of non-security experts. To build up comprehensive security analytics from this perspective, technical support for collaboration should be improved. Only with a well-developed infrastructure for collaboration between experts from different domains can the broadest possible protection against a wide variety of attack vectors be successful.

These problems form the starting point for the basic idea of our prototype. We aim to simplify the creation and processing of signatures (patterns), which can be used to detect attacks or at least indicators of compromise. This central concept is supported by an approach for visual programming, which is already established in education. With the help of visual programming, the entry barriers for complex systems can be successfully lowered [34, 35]. The objective is to make a complex, text-based syntax for defining patterns for attack detection easier to understand and use. To achieve this goal, we define the following requirements:

R1—Overview of currently deployed patterns: For users to get a quick overview of patterns that are currently already in use, the prototype must enable a corresponding display. All essential functions (such as editing a pattern) should be directly accessible from this overview view.

R2—Visual abstraction for complex pattern definition syntax: A selected visual programming approach should make the complex syntactic structure more accessible to users with little operational knowledge. It is essential that users can externalize their knowledge in a semantically simplified way. At the same time, the prototype has to ensure the correct, necessary syntax for the mechanism used for incident detection.

R3—Details for deployed patterns including situational context: If necessary, all details of a defined attack pattern should be available via the prototype. These details include the processing timestamps, the final pattern statement, and an insight into the activities or events associated with this pattern.

R4—Debugging mechanism for patterns: To promote an understanding of how the patterns work, the prototype should at least provide an easy way to debug the statements. Such debugging should clarify the relationships between individual events that have led to the triggering of the attack detection. In addition, it is also desirable that debugging can represent hierarchies of patterns of varying complexity.

R5—Centralized pattern storage and detection mechanism: To provide technical support for (remote) collaboration between different users, the prototype must centrally store and manage the defined signatures. The detection mechanism that uses the patterns must also be located in the center. Accordingly, architectures corresponding to a client-server structure should be aimed for.

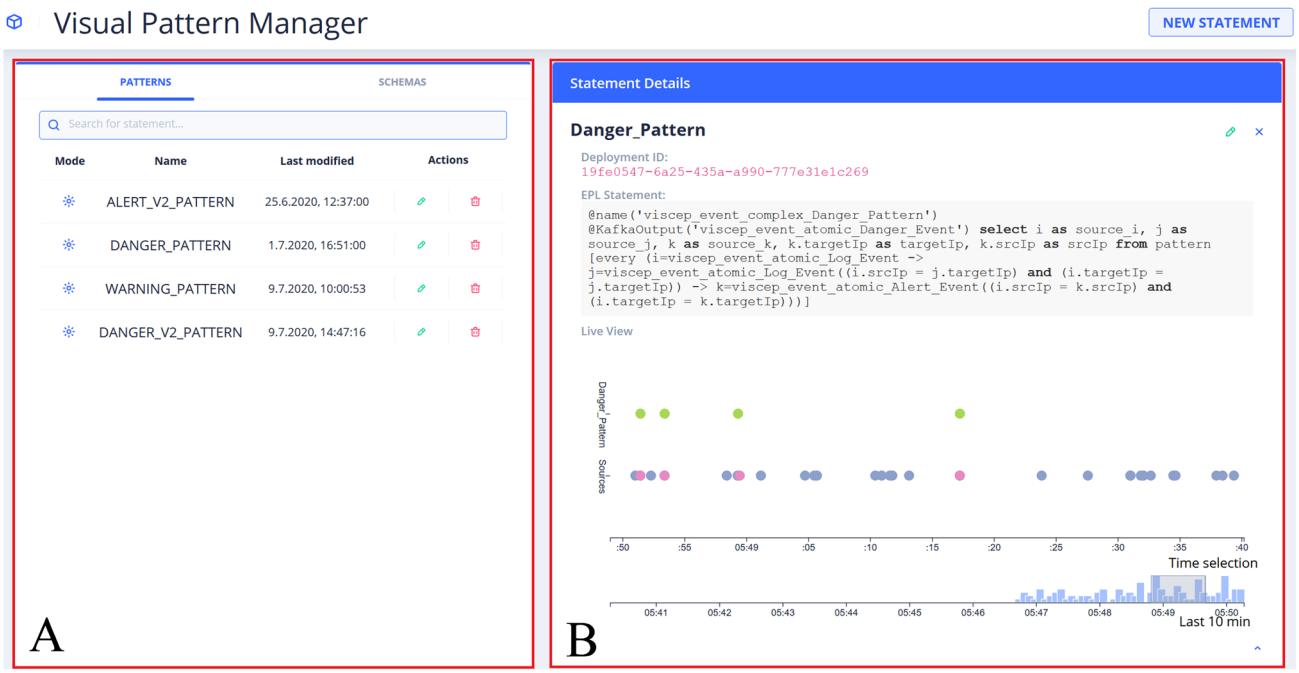

Fig. 7 Screenshot of the front end's landing page with a selected statement [12]

System Architecture

The prototype we developed is built on a client-server architecture, which is shown in Fig. 6. The server is the backend responsible for detecting indicators and identifying incidents based on predefined patterns and signatures. On the other hand, the frontend provides a user interface that enables the creation, editing, and debugging of these patterns. The entire system architecture of our prototype is based on open source technology. The source code of the application itself is also available as open source on GitHub[2].

Back End

In the following, the individual components of the back end are outlined, whereby their interconnection is shown in Fig. 6. In the backend, the actual rule-based event correlation takes place with the help of the Complex Event Processing Engine Esper[3] The actual events are provided by various data sources that reflect the current situation of the Incident Detection Lifecycle. With the help of Apache Kafka[4], a central message broker is provided that manages and passes on the events generated by the various backend components. Patterns created in the front end are persisted in the pattern storage (MongoDB[5]) to make them available to Esper on demand (cf. R5). An API Provider implements the connection between back end and front end using a modern GraphQL[6] interface.

Front End

The front end of our prototype consists of three basic views, which are embedded in an overarching user interface (UI). The UI is based on Angular.[7] The first view, the landing page, is divided into two components. These components are marked with two red boxes (A) and (B) in Fig. 7. The left component (A) provides an overview of all currently defined patterns and related details, such as the name of the pattern, the time of the last change of the pattern, and its current deployment mode (R1). This deployment mode indicates whether a pattern is still under development (i.e., whether work is currently being done on it) or whether it has already been integrated into the back end's incident detection operations. In addition, the pattern overview in component (A) can be used to initialize the editing of a pattern or to delete the corresponding pattern. By clicking on the "pencil" icon (i.e., editing a pattern), a user opens the current definition of the pattern in the *Visual Pattern Builder*, which is described in more detail in section "Visual Pattern Builder". A new

[2] https://github.com/Knowledge-based-Security-Analytics
[3] http://www.espertech.com/esper/.
[4] https://kafka.apache.org/.
[5] https://www.mongodb.com.
[6] https://graphql.org/.
[7] https://angular.io/.

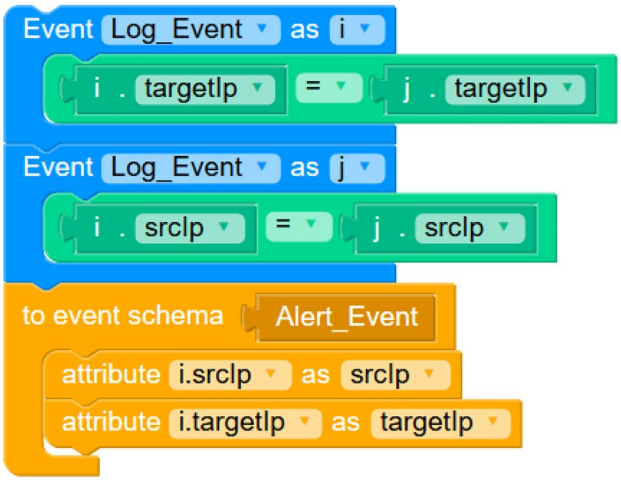

Fig. 8 Screenshot of EPL statement built with Blockly [12]

pattern is created via the "New Statement" button in the navigation bar. This action opens the Visual Pattern Builder without an already existing pattern definition, only with an empty editing area. In addition to the patterns, the overview also contains a tab for schemas. The event types defined there can be used to use the pattern. However, since their structure is very straightforward and event types can also be defined using the Visual Pattern Builder, we will focus on defining the patterns in the remainder of this section.

Respective for R3, the second component (B) from Fig. 7 of the landing page contains further information about a pattern selected in component (A). This includes the ID and the EPL statement, which formally describes the pattern and which is used in the pattern matcher. In addition to this information, component (B) presents a *Live Event Chart*, which provides a quick overview of the activities to be assigned to the pattern within the last 10 min. The bar chart in the lower part shows the entire time window (10 min) and the number of events registered to the pattern. The upper part of the event chart represents an interactively selectable time frame from the last minutes and the events generated by the pattern matcher after a match was identified within a set of source events and the source events themselves. Herein, circles with the same colors correspond to the same event type.

Visual Pattern Builder

This component of our prototype is used to create new statements or edit existing statements. For this purpose, we use the visual code editor Google Blockly[8]. Blockly has so far been used primarily in the educational environment, for example, to teach the basic principles and concepts of programming. The approach of Google Blockly is catchy and straightforward. It allows the definition of specific, logical building blocks, which the users can then assemble and parameterize. In the background, these blocks are compiled into executable source code. Blockly has proven its ability to lower entry barriers for novice users in many places. Therefore, we consider it suitable for abstracting the complex syntax and logical flow of the EPL expressions used within the pattern matcher (R2).

In our prototype, we implemented building blocks based on Google Blockly to create and edit Esper EPL expressions. Figure 8 shows a simple EPL statement defined with Blockly. The main components of these statements are event patterns (blue blocks), conditions (green blocks), and actions (yellow blocks). The pattern shown in Fig. 8 instructs the Pattern Matcher to emit an "Alert_Event" with the corresponding attributes after detecting two consecutive "Log_Event" instances with matching "srcIp" and "targetIp" attributes. An example of an Esper EPL expression generated by corresponding Google Blockly modeling can be seen in the gray box in component (B) in Fig. 7.

Please refer to our open-source implementation linked above for the full range of different Esper EPL statements supported. Among others, our implementation includes a logical combination of event sequences (including "and", "or", "not"), counted event sequences, and logical conditions. Although we cannot yet express all possible Esper EPL expressions as Blockly building blocks, the concept is promising so far. In subsequent iterations of our work, we expect to achieve near-complete coverage of Esper EPL.

Pattern Debugger

Using the arrow on the lower right side of component B, the *Patter Debugger* (Fig. 9) can be opened for the respective pattern. It allows testing of the created patterns by providing a detailed view of the event's data and its relationships (R4). As mentioned before, observables are assigned to an indicator and indicators to an incident in a hierarchical way. This hierarchy is visualized with the help of the pattern debugger to make relations easily recognizable. For displaying the hierarchy in a structured way, observables, indicators, and incidents are arranged next to each other in columns. The elements above or below in the tree are highlighted when hovering over them with the cursor to highlight the elements' hierarchical structure further.

The individual elements are represented as JSON. To maintain an overview, they are initially displayed in collapsed form. Only by selecting an element the complete JSON tree expands, whereby besides the overview, the option for displaying details is provided.

[8] https://developers.google.com/blockly

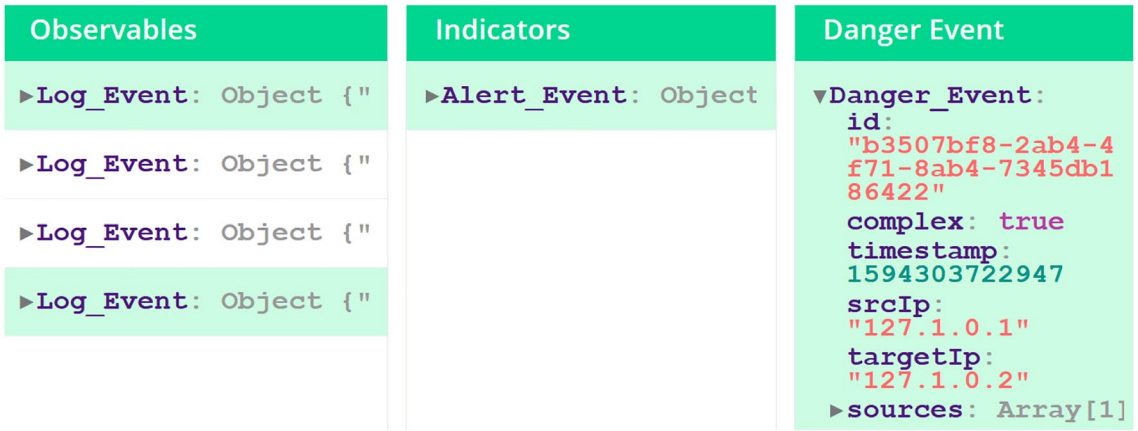

Fig. 9 Screenshot of the pattern debugger [12]

Discussion

To conclude the description of the prototype, it is discussed subsequently, emphasizing the implementation of the requirements and the approach to the underlying problems.

R1 is implemented on the landing page. An overview of all patterns is given here. The direct accessibility of the activities for the patterns is also available here in the form of action icons.

R2 is implemented with the help of Google Blockly. Using this technology makes it possible to create patterns without having to use complex text-based syntax. The visual programming approach additionally ensures that no erroneous patterns can be created. In our implementation, the syntax of the pattern is abstracted by the visual programming language while avoiding to cut the functionality of the pattern language. Additional research would be needed here to determine which level of abstraction is most appropriate.

R3 is implemented using a detailed view when a pattern was selected. The events that are affected by this pattern are visualized in the form of a live view to present the relationships in a comprehensible way.

R4 is implemented with the help of the pattern debugger. Within the debugger, the user has the possibility to highlight the events that have led to the triggering of an alarm. The respective events are hierarchically divided into Observables, Indicators, and Incident.

R5 is mainly implemented on the backend side. There, all created patterns are stored in the pattern storage to enable multiple users to work on them collaboratively. Furthermore, with the help of the Esper-based pattern matcher, event correlation is performed centrally.

The implemented requirements contribute to solving the two underlying problems, reducing the required K^{i}_{o} and enabling coll between S_n and S_e. The problem of reducing the needed K^{i}_{o} was solved with the help of the visual programming approach. This way, it is possible to create patterns without requiring in-depth expert knowledge of pattern syntax. Above all S_n is enabled to contribute its $K^{i}_{d(nonSec)}$. In addition, the complexity of pattern debugging has been reduced. The user does not have to work through various log files but can visualize the events in an easy-to-understand way. To not overwhelm the user, only a very abstract view of the events is given in the form of a life chart. However, if the user has the necessary expert knowledge, he can display the details of the events. If an even more comprehensive view is desired, the pattern debugger can be used, which shows the relationships between the individual events. The abstract representation of the events also facilitates int.

The reduction of the required K^{i}_{o} already contributes to enabling coll between S_n and S_e, as it reduces entry barriers especially for S_n and thus enables him to participate in creating detection patterns. This, for example, gives S_n the possibility to adapt rules created by S_e and enrich them with their $K^{i}_{d(nonSec)}$ and thus refine them. In addition, this is achieved, because patterns are stored in a central location, and all actors have access to them. The combination of int and ext is thus combined to allow coll.

The novelty of the presented model is the formal consideration of knowledge in the security analytics domain. Even though previous research certainly uses the concept of knowledge in this domain, there has not been a unified formal view of it. The formal definition thus creates a consistent view that enables the delimitation of future research. The social implications of the presented model demand a brief discussion as well. Closer integration of different types of knowledge results in cybersecurity gaining importance in multiple areas, which should benefit society as a whole. However, closer collaboration also creates societal dependencies that can bring both advantages and disadvantages.

Like any research work, the prototype presented has its limitations and points to future research potential. For example, it would be worth evaluating whether rule creation can be further simplified. Even if the underlying syntax is much more accessible through our visual approach, rule creation could be even more straightforward. Debugging can also be enhanced to provide even deeper insight into how the Pattern Matcher works. Furthermore, it needs to be empirically evaluated to what degree the prototype actually reduces the required K_o^i. This can be done in a user study that examines what effect coll has on detection rates.

Conclusions

This article presents and formalizes the concept of knowledge, its facets, and the concept of knowledge conversion in the context of security analytics. Building on this formalization, we present a model for knowledge-based security analytics based on the incident detection lifecycle. Our structuring and conceptualization makes it possible to raise the mostly inconsistent and informal descriptions to a formal and consistent level. With this contribution, we lay a sound foundation for future research in the field of security analytics.

Several sub-areas and activities within the knowledge-based SA model could be identified as not sufficiently considered in academic research. We presented a research prototype to demonstrate the first possible approach for externalizing human domain knowledge and collaboration between security experts and security novices. This prototype leverages the power of modern visual programming approaches to reduce the operational knowledge required to interact with security analytics systems, thereby lowering the barrier to entry for security novices. This also allows these domain experts to better provide their knowledge, which is especially important for incident detection in the CPS and IoT context in the form of signatures.

Although we were able to present a first research prototype that addresses the first open challenges in security analytics, there is still room for future research. First, we need to develop further technical support for collaboration between security experts and security innovators. Our prototype shows first possibilities here, but the approach needs to be improved together with users. A corresponding evaluation of the prototype to empirically confirm its suitability is also necessary. Furthermore, approaches are needed to integrate situational knowledge into SA better. Although initial approaches to this exist in the human-as-a-security-sensor environment, they must be improved and further developed.

Acknowledgements This research was partly supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy (BayStMWi), as part of the INSIST project.

Funding Open Access funding enabled and organized by Projekt DEAL. Not applicable.

Availability of data and materials Not applicable.

Code availability Github Repositories: https://github.com/Knowledge-based-Security-Analytics.

Declarations

Conflicts of interest The authors declare that they have no conflict of interest.

References

1. Schneier B. Secrets and lies: digital security in a networked world. 15th ed. Hoboken: Wiley; 2015.
2. Ben-Asher N, Gonzalez C. Effects of cyber security knowledge on attack detection. Comput Hum Behav. 2015;48:51–61. https://doi.org/10.1016/j.chb.2015.01.039.
3. Zimmermann V, Renaud K. Moving from a"human-as-problem''to a"human-as-solution''cybersecurity mindset. Int J Hum Comput Stud. 2019;131:169–87. https://doi.org/10.1016/j.ijhcs.2019.05.005.
4. Kendrick C, Frohnmaier M, Georges M. Audio-visual recipe guidance for smart kitchen devices. In: Proceedings of the fourth international conference on natural language and speech processing (ICNLSP 2021); 2021. pp. 257–61.
5. Loukas G. Cyber-physical attacks. Butterworth-Heinemann, Oxford. 2015. https://doi.org/10.1016/C2013-0-19393-2.
6. Dietz M, Vielberth M, Pernul G. Integrating digital twin security simulations in the security operations center. In: Proceedings of the 15th international conference on availability, reliability and security (ARES), pp. 1–9. ACM, New York. 2020. https://doi.org/10.1145/3407023.3407039.
7. Eckhart M, Ekelhart A. Towards security-aware virtual environments for digital twins. In: Proceedings of the 4th ACM workshop on cyber-physical system security—CPSS '18, pp. 61–72. ACM, New York. 2018. https://doi.org/10.1145/3198458.3198464.
8. Vielberth M, Bohm F, Fichtinger I, Pernul G. Security operations center: a systematic study and open challenges. IEEE Access. 2020;8:227756–79. https://doi.org/10.1109/ACCESS.2020.3045514.

9. Schneier B. Click here to kill everybody: security and survival in a hyper-connected world. 1st ed. New York: W.W. Norton & Company; 2018.
10. Chen TM, Sanchez-Aarnoutse JC, Buford J. Petri net modeling of cyber-physical attacks on smart grid. IEEE Trans Smart Grid. 2011;2(4):741–9. https://doi.org/10.1109/TSG.2011.2160000.
11. Geyer T, Rübenthaler J, Marschner C, von Hake M, Fabritius MP, Froelich MF, Huber T, Nörenberg D, Rückel J, Weniger M, Martens C, Sabel L, Clevert D-A, Schwarze V. Structured reporting using ceus li-rads for the diagnosis of hepatocellular carcinoma (hcc)-impact and advantages on report integrity, quality and interdisciplinary communication. Cancers. 2021;13:3. https://doi.org/10.3390/cancers13030534.
12. Böhm F, Vielberth M, Pernul G. Bridging knowledge gaps in security analytics. In: Proceedings of the 7th international conference on information systems security and privacy, pp. 98–108. SCITEPRESS— cience and Technology Publications, Online Streaming. 2021. https://doi.org/10.5220/0010225400980108.
13. Sallos MP, Garcia-Perez A, Bedford D, Orlando B. Strategy and organisational cybersecurity: a knowledge-problem perspective. J Intellect Cap. 2019;20(4):581–97. https://doi.org/10.1108/JIC-03-2019-0041.
14. Ackoff RL. From data to wisdom. J Appl Syst Anal. 1989;16:3–9.
15. Frické M. The knowledge pyramid: a critique of the dikw hierarchy. J Inf Sci. 2009;35(2):131–42. https://doi.org/10.1177/0165551508094050.
16. Davenport TH, Prusak L. Working Knowledge: how organizations manage what they know. Boston: Harvard Business School Press; 2000.
17. Nonaka I, Takeuchi H. The knowledge creating company. Oxford: Oxford University Press; 1995.
18. Fayyad U, Piatetsky-Shapiro G, Smyth P. From data mining to knowledge discovery in databases. AI Mag. 1996;17(3):37. https://doi.org/10.1609/aimag.v17i3.1230.
19. Sacha D, Stoffel A, Stoffel F, Kwon BC, Ellis G, Keim D. Knowledge generation model for visual analytics. IEEE Trans Visual Comput Graph. 2014;20(12):1604–13.
20. Polanyi M. The tacit dimension. Chicago: University of Chicago Press; 2009.
21. Chen M, Ebert D, Hagen H, Laramee RS, van Liere R, Ma K-L, Ribarsky W, Scheuermann G, Silver D. Data, information, and knowledge in visualization. IEEE Comput Graph Appl. 2009;1(29):12–9.
22. Wagner M, Rind A, Thür N, Aigner W. A knowledge-assisted visual malware analysis system: design, validation, and reflection of kamas. Comput Secur. 2017;67:1–15. https://doi.org/10.1016/j.cose.2017.02.003.
23. Jaeger L. Information security awareness: literature review and integrative framework. In: Bui, T. (ed.) Proceedings of the 51st Hawaii International Conference on System Sciences. Hawaii International Conference on System Sciences, Honolulu. 2018. https://doi.org/10.24251/HICSS.2018.593.
24. Vasileiou I, Furnell S. Personalising security education: factors influencing individual awareness and compliance. In: Information systems security and privacy. communications in computer and information science, vol. 977, pp. 189–200. Springer, Cham. 2019. https://doi.org/10.1007/978-3-030-25109-3_10.
25. Ponsard C, Grandclaudon J. Guidelines and tool support for building a cybersecurity awareness program for smes. In: Information systems security and privacy. Communications in computer and information science, vol. 1221, pp. 335–357. Springer, Cham. 2020. https://doi.org/10.1007/978-3-030-49443-8_16.
26. Wang X, Jeong DH, Dou W, Lee S-W, Ribarsky W, Chang R. Defining and applying knowledge conversion processes to a visual analytics system. Comput Graph. 2009;33(5):616–23. https://doi.org/10.1016/j.cag.2009.06.004.
27. Federico P, Wagner M, Rind A, Amor-Amorós A, Miksch S, Aigner W. The role of explicit knowledge: a conceptual model of knowledge-assisted visual analytics. In: Proceedings of the IEEE conference on visual analytics science and technology (VAST). 2017.
28. Thalmann S, Ilvonen I. Why should we investigate knowledge risks incidents? Lessons from four cases. In: Bui, T. (ed.) Proceedings of the 53rd Hawaii International Conference on System Sciences. Hawaii International Conference on System Sciences, Honolulu. 2020. https://doi.org/10.24251/HICSS.2020.607.
29. Mahmood T, Afzal U. Security analytics: big data analytics for cybersecurity: a review of trends, techniques and tools. In: 2013 2nd national conference on information assurance (NCIA), pp. 129–134. IEEE, New York. 2013. https://doi.org/10.1109/NCIA.2013.6725337.
30. Menges F, Pernul G. A comparative analysis of incident reporting formats. Comput Secur. 2018;73:87–101. https://doi.org/10.1016/j.cose.2017.10.009.
31. National institute of standards and technology: framework for improving critical infrastructure cybersecurity, Version 1.1 2018. 2021. https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf. Accessed 14 Sep 2021.
32. Vielberth M, Englbrecht L, Pernul G. Improving data quality for human-as-a-security-sensor, a process driven quality improvement approach for user-provided incident information. Inf Comput Secur. 2021;2021:5.
33. Vielberth M, Menges F, Pernul G. Human-as-a-security-sensor for harvesting threat intelligence. Cybersecurity. 2019;2:1. https://doi.org/10.1186/s42400-019-0040-0.
34. Chao P-Y. Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. Comput Educ. 2016;95:202–15. https://doi.org/10.1016/j.compedu.2016.01.010.
35. Sáez-López J-M, Román-González M, Vázquez-Cano E. Visual programming languages integrated across the curriculum in elementary school. Comput Educ. 2016;97:129–41. https://doi.org/10.1016/j.compedu.2016.03.003.