

# Bridging Knowledge Gaps in Security Analytics

Fabian Böhm<sup>a</sup>, Manfred Vielberth<sup>b</sup> and Günther Pernul

*Chair of Information Systems, University of Regensburg, Germany*

**Keywords:** Security Analytics, Domain Knowledge, Visual Analytics, Security Awareness.

**Abstract:** In a cyber-physical world, the number of links between corporate assets is growing and infrastructures are becoming more complex. This and related developments significantly enlarge the attack surface of organizations. Additionally, more and more attacks do not exploit technical vulnerabilities directly but gain a foothold through phishing or social engineering. Since traditional security systems prove to be no longer sufficient to detect incidents effectively, humans and their specialized knowledge are becoming a critical security factor. Therefore, it is vital to maintain an overview of the cybersecurity knowledge spread across the entire company. However, there is no uniform understanding of knowledge in the field of security analytics. We aim to close this gap by formalizing knowledge and defining a conceptual knowledge model in the context of security analytics. This allows existing research to be better classified and shows that individual areas offer much potential for future research. In particular, the collaboration between domain experts but also between machines and employees could enable the exploitation of previously unused but crucial knowledge. For example, this knowledge is of great value for defining security rules in current security analytics systems. We introduce a proof of concept implementation using visual programming to showcase how even security novices can easily contribute their knowledge to security analytics.

## 1 INTRODUCTION


Humans are often considered the weakest link in cybersecurity (Schneier, 2015). However, they are also an invaluable asset as their domain knowledge is essential for any modern Security Analytics<sup>1</sup> (SA) method (Ben-Asher and Gonzalez, 2015; Zimmermann and Renaud, 2019). We argue that these methods could benefit significantly from the knowledge of non-security domains but mainly rely solely on security experts' knowledge to decide whether indicators are malicious incidents or benign activities.


This issue becomes apparent when looking at the Internet-of-Things (IoT) and ubiquitous Cyber-Physical Systems (CPSs). They lead to an increasing connectedness of organizations' internal and external assets. Besides an already skyrocketing number of cyber attacks, this significantly increases the attack surface for cyber-physical attacks. These explicitly exploit the linkage between cyber systems and physical systems within CPSs to do physical harm to machines

or even humans (Loukas, 2015). Detecting and mitigating these attacks poses a challenge to existing security measures. They need to monitor assets directly connected to cyberspace like firewalls and CPSs (full-blown manufacturing lanes). A general problem with CPSs is that current security means, e.g., Security Information and Event Management Systems (SIEMs) embedded within Security Operations Centers, lack both knowledge and abilities to protect them effectively (Dietz et al., 2020; Eckhart and Ekelhart, 2018).

Although security experts can make well-informed decisions about incidents in cyberspace, they lack the knowledge about the physical domain to decide whether, e.g., a turbine is acting as expected (Schneier, 2018). However, engineers and the respective staff have this knowledge to distinguish normal turbine behavior from anomalous actions but lack the knowledge to contribute to SA (Eckhart and Ekelhart, 2018).

This mismatch reduces organizations' ability to implement cohesive SA methods that can reliably detect both cyber and cyber-physical attacks with the respective indicators. Therefore, the domain knowledge of engineers and the alike need to be integrated into security operations to allow effective incident detection even for physical assets (Chen et al., 2011).

<sup>a</sup>  <https://orcid.org/0000-0002-0023-6051>

<sup>b</sup>  <https://orcid.org/0000-0002-1119-4715>

<sup>1</sup>As no widely accepted definition of Security Analytics exists, we interpret this term as a set of methods to identify attacks and threats through data analysis proactively.

In this work, we make a two-fold contribution to tackle this open issue. We first define the different types of knowledge within cybersecurity and formalize a model of knowledge-based SA. This is a new contribution as no security-specific notions of knowledge have been defined yet and it enables future research to identify and address more open challenges in this field. Our second contribution specifically addresses missing externalization of domain knowledge as one issue within knowledge-based SA by introducing a research prototype allowing experts to make their knowledge available.

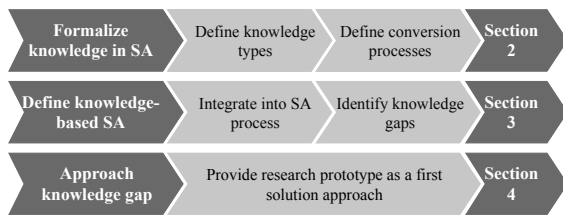


Figure 1: Schematic of this work's structure.

The remainder of this work is structured according to Figure 1. Before we can bridge knowledge gaps within SA efficiently, we need to formally describe relevant notions of knowledge and conversion processes for Security Analytics in Section 2. This allows us and any future work to have a well-defined, precise vocabulary. We then integrate this vocabulary into the Incident Detection Process to form a cohesive picture of what we call knowledge-based SA within Section 3. The resulting model reveals several knowledge gaps in current SA approaches that are not yet appropriately addressed. Thus, we present a research prototype in Section 4 showcasing a possible approach of the integration of 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 drastically improve cybersecurity. Finally, Section 5 concludes our work and points out possible directions for future work.

## 2 KNOWLEDGE WITHIN SECURITY ANALYTICS

In this section, we provide an in-depth view of knowledge within modern SA. Therefore, we establish a formal understanding of different types of knowledge and knowledge conversions within this section. Sallos et al. (Sallos et al., 2019) stress the importance of knowledge in cybersecurity on a high-level,

management-oriented view. We aim to add a more formal view focusing on the implications of integrating knowledge into security measures.

### 2.1 Knowledge Types

Within the scientific literature, there are numerous competing definitions of knowledge and different notions of knowledge. At an abstract level, we follow Davenport's definition, describing knowledge as a mix of experience, intuition, values, contextual information, and expert insight (Davenport and Prusak, 2000). In the context of the data-information-knowledge-wisdom (DIKW) hierarchy, knowledge describes the application of data and information to provide answers to "How"-questions (Ackoff, 1989).

However, the concept of knowledge is not bound to humans. Early research shows that within organizations, knowledge gets transcribed into documents or files (Nonaka and Takeuchi, 1995). Following this path, it becomes apparent that ICT can hold a specific type of knowledge different from human knowledge, especially when it comes to any type of automated data analysis (Fayyad et al., 1996; Sacha et al., 2014). Therefore, it is an accepted definition within ICT research to differentiate between two main types of knowledge: explicit knowledge and tacit knowledge. In the following, we will elaborate on those notions and put them into the context of SA.

#### 2.1.1 Explicit Knowledge

Explicit Knowledge  $K^e$  can be defined as machine knowledge that can be read, processed, and stored by machines (Nonaka and Takeuchi, 1995). We differentiate two different forms of explicit knowledge within Security Analytics from the main incident detection methods. For anomaly-based detection mechanisms, machine-readable knowledge is machine learning models and alike. However, signature-based security analytics holds explicit knowledge in the form of signatures and rules to detect indicators of compromise. Another SA-specific example for  $K^e$  is Cyber Threat Intelligence (CTI), as it describes attributed incidents in a structured way that can be exchanged and utilized by machines.

#### 2.1.2 Implicit Knowledge

In general, Tacit Knowledge can only be held by humans and is very specific to the individual (Polanyi, 2009). Although "tacit knowledge" is the more widespread term, we use implicit knowledge  $K^i$  throughout this work as this more clearly indicates the difference between machine and human knowledge.

Implicit knowledge is gained by linking new insights and prior knowledge, which can be further subdivided into the notions of domain and operational knowledge (Chen et al., 2009). Domain knowledge describes what users know about a specific context (i.e., domain) like security or engineering (Ben-Asher and Gonzalez, 2015; Eckhart and Ekelhart, 2018). Operational knowledge, in contrast, is the ability of a human to interact with a specific system. However, in the context of SA, we consider this differentiation too vague. To describe the problem at hand concisely, a fine-granular and contextualized view on  $K^i$  is necessary.

One straight-forward contextualization of  $K^i$  in SA is the one of operational knowledge  $K_o^i$ . Employees with operational knowledge can operate an organization's security systems. This might range from having the experience to create signatures for a SIEM system or tuning models for anomaly- and behavior-based SA mechanisms.

In Equation 1 we define the domain knowledge relevant for SA as a combination of the two sub-types  $K_{d(sec)}^i$  and  $K_{d(nonSec)}^i$ . The first one encapsulates any security-related domain knowledge that is mostly held by security domain experts. Elements of security-related domain knowledge are safety and security aspects from a cybersecurity perspective, like firewall rules, suspicious network connections, or unauthorized digital access to information. We perceive this type of domain knowledge to be already integrated to a good degree in SA means (Wagner et al., 2017). However,  $K_{d(nonSec)}^i$  as non-security domain knowledge, like manufacturing or engineering knowledge, is not yet sufficiently integrated into SA, although being relevant to detect incidents on cyber-physical systems (Dietz et al., 2020). As examples of non-security domain knowledge serve security aspects from an engineering view like expected RPMs of a power turbine or temperature in a blast furnace.

In the domain of SA, we consider an additional, new type of implicit knowledge: situational knowledge  $K_s^i$ . This type encompasses the concept of security awareness of an organization's employees (Jaeger, 2018; Vasileiou and Furnell, 2019). Any of them can perceive unusual events or suspicious behavior in the real-world. This ranges from a received email, which probably is a phishing attempt, to a private USB drive plugged into a company computer. Having situational security knowledge (e.g., after security awareness trainings (Ponsard and Grandclaudon, 2020)), the employee is able to derive that the email or the USB drive might pose a threat to the company. No specific security domain knowledge ( $K_{d(nonSec)}^i$ ) about the company's SA is necessary

for these types of conclusions.

All of these different implicit knowledge types are relevant for SA to detect both cyber and cyber-physical incidents. This allows us to aggregate our view on implicit knowledge within SA as a combination of the three previously introduced sub-types of  $K^i$  (Eq. 2).

$$K_d^i = K_{d(sec)}^i \cup K_{d(nonSec)}^i \quad (1)$$

$$K^i = K_d^i \cup K_s^i \cup K_o^i \quad (2)$$

$K_s^i$ , as well as  $K_d^i$ , need to be made available for cohesive security operations. The operational knowledge  $K_o^i$  in the context of SA poses an entry barrier for this integration. Having the necessary  $K_o^i$  would allow any employee, for example, to define a signature for a SIEM system. A user's knowledge in this context can be defined as instances of the different types of  $K^i$  as shown in Equation 3. Based on their different knowledge sets we differentiate two types of security-related users: security experts  $S_e$  (Eq. 4) and security novices  $S_n$  (Eq. 5). Differences between these employee groups are the different forms of domain knowledge and the lack of operational knowledge for security novices. This sheds light on a dichotomy within SA as security experts have the necessary operational knowledge  $K_o^i$  but lack, e.g., engineering domain knowledge  $K_{d(nonSec)}^i$ , which is relevant for SA to identify cyber-physical attacks.  $S_n$  might have this engineering knowledge but lack  $K_o^i$  analogously.

$$k_{d(nonSec)}^i, k_{d(sec)}^i \in K_d^i, k_s \in K_s^i, k_o \in K_o^i \quad (3)$$

$$S_e = \{k_{d(sec)}^i, k_s^i, k_o^i\} \quad (4)$$

$$S_n = \{k_{d(nonSec)}^i, k_s^i\} \quad (5)$$

## 2.2 Knowledge Conversion

The transfer between explicit to implicit knowledge and vice versa is formalized by Nonaka and Takeuchi, who introduced four knowledge conversion processes (Nonaka and Takeuchi, 1995). Several research domains have been picking those up to formalize the interaction between computers and humans. Naturally, the domain of visualization and human-computer interaction are lending heavily from these concepts (Wang et al., 2009; Federico et al., 2017; Wagner et al., 2017). However, we also think that the area of SA can benefit from them as it is relying on automated detection processes (explicit knowledge), but also the expertise of different human experts (implicit knowledge) is necessary for effective security operations. Therefore, we define the following four knowledge conversion processes for SA:

1. **Internalization** (*int*). This process describes explicit knowledge being made available for humans so that they can perceive it using their operational knowledge and transform it into security-related domain knowledge (Eq. 6). The amount and effectiveness of the knowledge conversion are heavily dependent on the user's  $k_o^i$ . This dependence on  $k_o^i$  is indicated in the equation as the operational knowledge is annotated as the catalyst of the knowledge conversion. This notation is adapted from formal descriptions of chemical reactions. An example of effective internalization of  $K_e$  in the context of SA is any form of its visual display.
2. **Externalization** (*ext*). Through externalization, both  $K_d^i$  and  $K_s^i$  are formalized to be read, stored, and processed by computers (Eq. 7). Again, this process is enabled by and dependent on  $K_o^i$ . Various ways help to realize this conversion in SA. Examples are the direct tuning of model parameters ( $K_e$ ) by experts and the definition of rules for signature-based analyses. Another example is the direct access for experts to CTI and a way to actively edit the accessed information. Externalization is also important as the loss of  $K^i$  through. For example, security analysts' retirement poses a risk to organizations as occurred incidents show (Thalmann and Ilvonen, 2020).
3. **Combination** (*comb*). A knowledge conversion process representing the combination of two or more explicit knowledge bases (Eq. 8). The exchange of CTI information between different actors and its utilization in their respective SA operations can be characterized as a combination process.
4. **Collaboration** (*coll*). Humans are working together and therefore combining their  $K^i$  (Eq. 9). Collaboration is a hard process to formally capture and define. However, in the context of SA, we see this knowledge conversion as a more technologically supported process. Employees, for example, work together on the correlation of indicators to figure out whether the indicator marks an ongoing attack or not. Therefore, they might use an analysis tool designed specifically for this purpose, supporting the collaboration. During their collaborative analysis, they are learning from each other and convert knowledge in the sense of *coll*.

$$int : (K_e \xrightarrow{K_o^i} K_{d(sec)}^i) \quad (6)$$

$$ext : (K_d^i \cap K_s^i \xrightarrow{K_o^i} K_e) \quad (7)$$

$$comb : K_e \mapsto K_e \quad (8)$$

$$coll : K^i \mapsto K^i \quad (9)$$

### 3 KNOWLEDGE-BASED SECURITY ANALYTICS

After having established a formal understanding of knowledge types and conversions relevant for SA in the previous section, we set out to integrate these notions of knowledge into SA's core process. The detection of incidents, i.e., attacks on any asset of a company, is one of the main functionalities of SA (Mahmood and Afzal, 2013). A cohesive approach to this process requires comprehensive data collection and the integration of any available knowledge. In this section, we elaborate on our model of knowledge-based SA and the crucial role of knowledge in this context. Finally, we identify process steps needing more attention from research.

#### 3.1 Knowledge Model

Menges and Pernul define a well-thought incident detection process (Menges and Pernul, 2018). We integrate the knowledge types and conversions from the previous sections to show the relationships between incident detection and knowledge of both humans and machines in Figure 2. The gray boxes in the figure mark the components of the original incident detection process.

The first phase of the incident detection process, *Data Collection*, deals with collecting raw *Data* that is generated by *Situations*. A situation is anything "that happens" within a company, either in the physical world or in cyberspace. This data is normalized (and sometimes standardized) and then forms *Observables* (Menges and Pernul, 2018). Note that observables only cover normalized facts of what happened within the collected raw data but are not attributed yet, so they do not answer why something happened or who is responsible. Observables serve as input for the second phase of the process, which is the actual *Incident Detection*. Within this phase, the observables are analysed to detect *Indicators*, often related to as *Indicators of Compromise* (IoCs). These point out possibly malicious or at least suspicious activities within the observables. However, IoCs can also just stem from unusual but benign behavior recorded within the



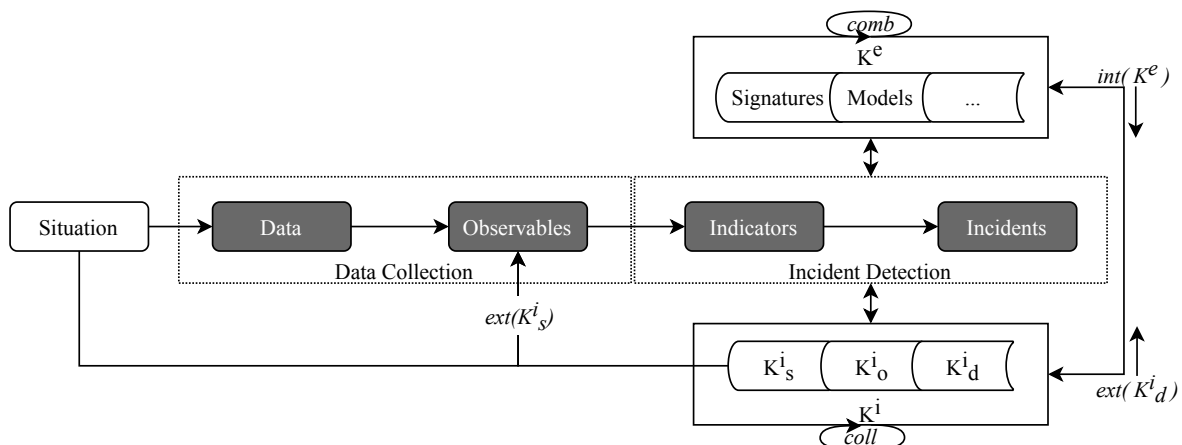


Figure 2: Model of knowledge-based SA based on the Incident Detection Process (Menges and Pernul, 2018).

observables. Therefore, indicators need to be correlated to identify actual *Incidents*.

Deriving and detecting indicators within the vast amount of observables is a considerably challenging task, which relies heavily on automated analysis processes because of the sheer number of available observables (Mahmood and Afzal, 2013). These processes use explicit knowledge  $K^e$  as signatures and rules for signature-based detection but also in the form of models for behavior-based methods. Therefore,  $K^e$  plays a significant role in incident detection. Nevertheless, there are major drawbacks with only integrating  $K^e$  into automated detection. First of all,  $K^e$  in the form of *Signatures* only can detect already known indicators. Additionally, behavior-based *Models* tend to produce many false positives. Human domain experts can help to overcome both shortcomings. On the one hand, they can explore available observables to identify new indicators, and on the other hand, they leverage their domain knowledge to decide whether an indicator imposes a malicious action or not. Therefore, the integration of  $K^i_d$  in this step is beneficial and a prerequisite for effective SA.

The next process step, identifying actual incidents within Indicators, needs even more integration of  $K^i$ . This is mainly because  $K^e$  in this step can only identify previously known attacks where attack patterns are already defined. When new attacks occur,  $K^e$  can barely do more than detect indicators. It cannot identify the incident in its full scope or complexity.  $K^i_d$  is needed for this task as experts can differentiate between malicious and benign indicators and correlate them.

Besides this incorporation of  $K^e$  and  $K^i$  into the *Incident Detection* phase we also indicate several knowledge conversion processes within Figure 2. Those are the conversions we identify being rele-

vant and necessary for holistic, effective SA covering both cyber and cyber-physical incidents. We introduced the indicated processes  $int(K^e)$  (Eq. 6),  $comb$  (Eq. 8), and  $coll$  (Eq. 9) introduced in the previous Section 2.2. Thus, we will focus the two remaining processes  $ext(K^i_s)$  and  $ext(K^i_d)$  hereinafter. They are instances of  $ext$  (Eq. 7) but have not been defined in more detail yet.

The externalization of situational knowledge  $ext(K^i_s)$  essentially allows employees to transform events they have observed or experienced into observables. Thus, the semantic information encapsulated in those can be used in the following stages of the incident detection process. Leveraging  $ext(K^i_s)$  supplements any traditional data collection because many of the aspects of targeted attacks remain unseen within automatically collected data, for example, social engineering or physical access attempts. Therefore, this knowledge conversion turns any employee (both  $S_e$  and  $S_n$ ) into a valuable source when they, for example, report a phone call trying to find out their access credentials. This process does not rely on domain knowledge but on a more general knowledge, which can be achieved through situational awareness.  $ext(K^i_s)$  is particularly relevant as it is the only way to cover the full scope of possible attack vectors, which could not be achieved by  $S_e$  alone.

The process  $ext(K^i_d)$  essentially comprises both  $ext(K^i_{d(sec)})$  and  $ext(K^i_{d(nonSec)})$ . It describes the interaction of humans with actual analysis methods backed by  $K^e$ . The goal of this is to make  $K^i_d$  available for automated analysis methods to improve incident detection. In the age of cyber-physical systems, domain knowledge, especially  $ext(K^i_{d(nonSec)})$ , is widely scattered across organizations while at the same time, all of this knowledge is necessary for cohesive SA. Therefore,  $ext(K^i_d)$  is crucial as it allows

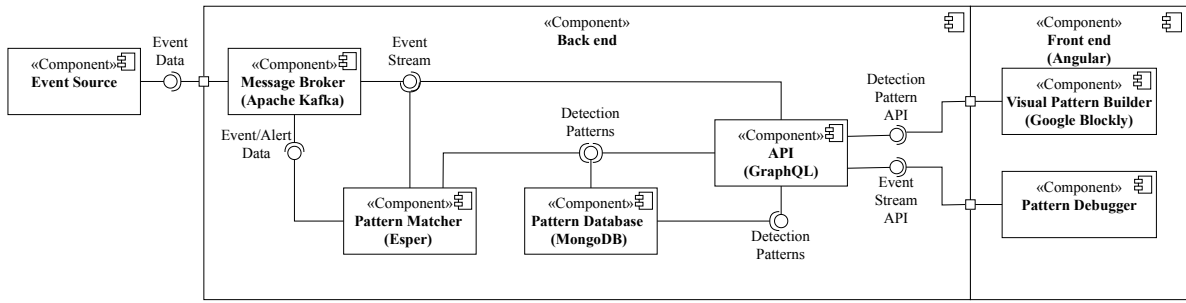


Figure 3: Component diagram of the architecture for visual collaborative pattern definition.

human knowledge to be transferred into new SIEM correlation rules, attack signatures, or improved behavioral models for an organization’s assets.

### 3.2 Knowledge Gaps

Building on the formal definitions and the model from the previous sections, we specify what we call SA’s dichotomy. As Equations 4 and 5 show, the main reason for the dichotomy is the security novice’s lack of operational knowledge. This leads to three knowledge gaps limiting current SA operations within organizations and are not yet sufficiently addressed by research. In this section, we provide a more detailed view of these pressing issues to be solved in order to allow the integration of relevant knowledge from all employees into cohesive SA operations:

1.  $\text{ext}(K_s^i)$ : The first problem needing more attention from academia is the externalization of  $K_s^i$ . A key difficulty with this lies in creating means to externalize the knowledge, i.e., how to enable security novices  $S_n$  to turn their situational knowledge  $K_s^i$  into *Observables* as indicated in Figure 2. Initial research approaches addressing this problem consider employees as a kind of security sensor, which allows them to provide observations and perceptions from the real world as input for SA (Vielberth et al., 2019; Heartfield and Loukas, 2018). Although the first concepts to establish this knowledge conversion exist, this problem is far from being completely solved.
2.  $\text{ext}(K_{d(\text{nonSec})}^i)$ : The second problem stems from the lack of  $K_o^i$  of security novices  $S_n$  like engineers for externalizing their  $K_d^i$ . Therefore, ways to lower this entry barrier have to be researched. For example, engineers could contribute heavily in defining rules for a SIEM system as part of an organization’s SA. The knowledge of engineers in this context is relevant as they know better how, e.g., an electric turbine should not behave. This is needed for detecting indicators of attacks tar-

getting the turbine itself or its cyber-components, i.e., the complete cyber-physical system. However, engineers are unlikely to be able to define SIEM correlation rules in a possibly cumbersome syntax. Therefore,  $\text{ext}(K_{d(\text{nonSec})}^i)$  must be simplified for  $S_n$ . This process lacks attention from research as there are no concepts available yet how to achieve this in SA.

3. **coll**: Collaboration, especially between  $S_e$  and  $S_n$ , is crucial to bring together the scattered domain knowledge into a centralized knowledge base for SA. Any solution for the previous two issues in combination with means for internalization *int* can support collaboration. Although collaboration is a central part of modern organizations, it has not yet been focused on in SA research in terms of bringing together security experts and novices.

## 4 RESEARCH PROTOTYPE

As mentioned in Section 3.2 there is no existing work that directly addresses the processes  $\text{ext}(K_{d(\text{nonSec})}^i)$  and *coll*. In the following, we, therefore, present a research prototype of signature-based incident detection, which supports those two knowledge conversions and is conceptualized along with the model of knowledge-based SA (see Figure 2). To detect indicators and incidents, we apply complex event processing based on a pattern hierarchy. This hierarchical approach allows observing indicators based on both observables and other indicators. Additional patterns can be used to identify actual incidents by correlating IoCs. The patterns, i.e., signatures required for this purpose, are interpreted as  $K^e$  in the context of SA and are accessible for humans using the prototype. For the sake of clarity, we use the general term of event whenever it is not necessary to distinguish between observable, indicator, or incident.

Overall, our prototype aims at minimizing the required  $K_o^i$  for  $\text{ext}(K_{d(\text{nonSec})}^i)$  by providing centralized,

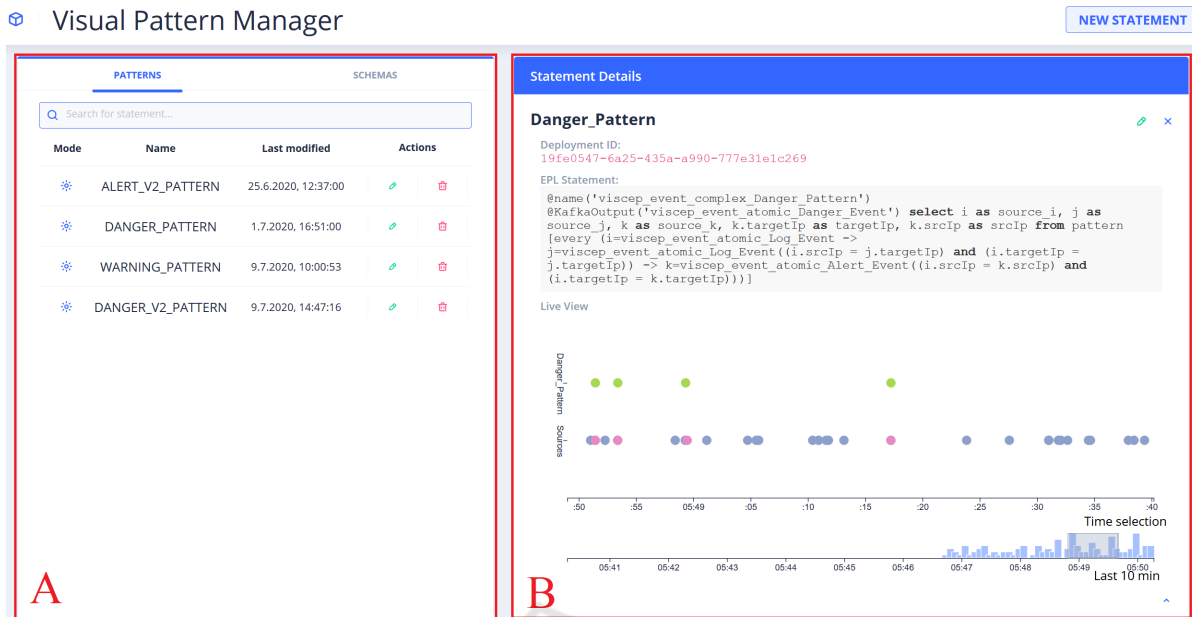


Figure 4: Screenshot of the front end’s landing page with a selected statement.

visual access to the  $K^e$ . Additionally, it supports *coll* among humans. Its basic idea is to simplify the creation and editing of patterns as it detaches them from a complex, text-based syntax. This central concept is enabled by visual programming, which has proven its ability to lower entry barriers of complex systems, especially in the context of education (Chao, 2016; Sáez-López et al., 2016). This makes complex, text-based syntax easier to understand. Thus, the prototype aims to meet the following requirements:

- **Reduce the Needed  $K_o^i$  for  $ext(K_d^i)$ .** Although  $K_o^i$  is necessary for both *int* and *ext*, it serves no other, security-specific purpose. Therefore, the required  $K_o^i$  should be reduced, especially for  $S_n$ . They are not familiar with security solutions such as SIEM systems and to harvest their knowledge for security purposes, it is necessary to keep the entry barriers ( $K_o^i$ ) as low as possible. Concerning the notation of  $K_o^i$  being the catalyst of knowledge conversions, we aim to reduce the catalyst needed to initiate the conversion.
- **Enable *coll* between  $S_e$  and  $S_n$ .** Security experts have the knowledge about possible attack vectors, however, the knowledge needed to adapt the attack vectors to a specific context often lies within the domain of non-security experts. Thus, collaboration is necessary for identifying as many attack vectors as possible.

## 4.1 System Architecture

As shown in Figure 3, the prototype’s architecture is divided into front end and back end. The back end is responsible for recognizing indicators and incidents based on predefined patterns and the front end provides a user interface allowing to create, edit, and debug these patterns. Please note that the architecture is completely based on open source technologies and the source code is available for everyone on GitHub<sup>2</sup>.

### 4.1.1 Back End

Since the back end is indispensable for the application’s functionality, but the actual contribution of the paper lies in the front end, we will only give a quick and superficial view of it. In the back end, indicators and incidents within an event stream are detected with the help of complex event processing based on Esper<sup>3</sup>. Events are made available by several *Data Sources* and provided by the *Event Broker* based on Apache Kafka<sup>4</sup> as an event stream. The patterns formulated using the Esper Event Processing Language<sup>5</sup> (EPL) are created in the front end and persisted in the *Pattern Storage* (MongoDB<sup>6</sup>). Between the front end and the back end, an *API Provider*

<sup>2</sup> <https://github.com/Knowledge-based-Security-Analytics>

<sup>3</sup> <http://www.esper.tech/esper/>

<sup>4</sup> <https://kafka.apache.org/>

<sup>5</sup> <https://esper.esper.tech/release-5.2.0/>

<sup>6</sup> <https://www.mongodb.com/>

is implemented as a modern GraphQL<sup>7</sup> API providing access to the event stream and allowing to create, update, and delete event patterns.

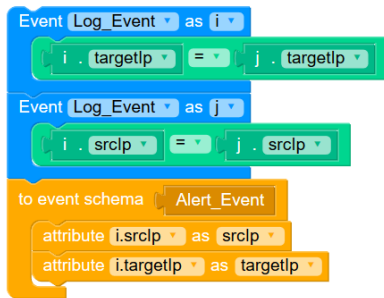


Figure 5: Screenshot of EPL statement built with Blockly.

#### 4.1.2 Front End

Our front end essentially consists of three main views embedded in an Angular<sup>8</sup>-based User Interface (UI). The landing page is divided into two main components highlighted with red boxes (A) and (B) in Figure 3. On the left, component (A) provides an overview of the defined patterns, including meta-data as the name, time of the last modification, and the deployment mode. The deployment mode indicates whether a pattern is still being developed or has already been deployed into operations. Additionally, the pattern overview allows to start the editing of a pattern or to delete it. Clicking on the “Pencil”-action (i.e. edit a pattern) brings up the *Visual Pattern Builder* as described in Section 4.2 with the current definition of the pattern. The *Visual Pattern Builder* can also be started to define a new pattern via the “New Statement”-button in the navigation bar. Please note that the overview also has a tab indicating *Schemes*. These define the event types that can be used to describe a pattern, but as they are very straight-forward, we will focus on patterns.

The second component (B) of the landing page contains additional information about a pattern. Besides its ID and EPL statement as used in the *Pattern Matcher* this component displays a *Live Event Chart* for a quick overview of the pattern’s activities within the last ten minutes. The bar chart feature on the bottom of the *Live Event Chart* shows the full time window and the number of events registered on the pattern. The upper part of the event chart displays an interactively selectable time frame within the last ten minutes and both the events generated by the *Pattern Matcher* after having identified a match on specific sets of source events and the source events them-

selves. Circles of the same color correspond to the same event type.

## 4.2 Visual Pattern Builder

This component is used to create new statements or edit existing ones. The visual code editor Google Blockly<sup>9</sup> is used for this purpose. Google Blockly is primarily used in the education context to teach, for example, the basic principles of programming. It allows defining specific logical building blocks that can be snapped together and parametrized by the user. Internally, these blocks are compiled into working source code. Blockly has proven its feasibility in lowering entry barriers for novice users. Therefore, it is a good candidate for abstracting the rather complex syntax and logical flow of the EPL used within the *Pattern Matcher*.

In our prototype, we implemented blocks based on Google Blockly that allow Esper EPL statements’ construction. The main parts of these statements are event patterns (blue blocks), conditions (green blocks), and actions (yellow blocks). Figure 5 shows a simple statement built with Blockly. The displayed statement instructs the *Pattern Matcher* if it sees two instances of “Log\_Event” followed by each other and containing the same “srcIp” and “targetIp” attribute, it should produce an “Alert\_Event” with the respective attributes. An example of a resulting Esper EPL statement is displayed in the grey box in component (B) in Figure 4.

Please refer to our implementation for the bandwidth of different EPL structures that we already implemented in Blockly. It includes, besides others, logical combinations of event sequences (including “and”, “or”, “not”), counted event sequences and logical conditions. Although we are not yet capable of representing all possible EPL structures, the concept and implementation are very promising, and we plan to come closer to full coverage of the Esper EPL within future iterations of our work.

## 4.3 Pattern Debugger

Clicking on the small arrow on the bottom right side of component (B) brings up the *Pattern Debugger* for the respective pattern. It allows testing the created patterns by providing a detailed view of the events’ data and their relationships. As stated before, one or multiple observables can lead to indicators and one or multiple indicators can lead to incidents. This hierarchical order is also represented within the pattern

<sup>7</sup> <https://graphql.org/>

<sup>8</sup> <https://angular.io/>

<sup>9</sup> <https://developers.google.com/blockly>



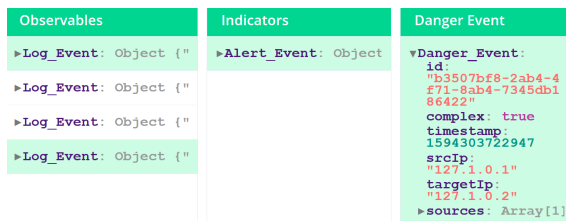


Figure 6: Screenshot of the Pattern Debugger.

debugger. The pattern to debug either detects an indicator or an incident. Subsequently, this view displays three columns, as shown in Figure 6. The first column shows all observables, which are the source for any pattern related to the selected one. The second column shows all indicators, which lead to the debugged pattern and the third column shows the output (indicators or incidents) of the pattern itself.

The respective observables, indicators, and incidents are shown in a JSON tree structure. Initially, each JSON tree is shown collapsed, where a user can only see the name and a small amount of information. If a more detailed investigation of certain events is necessary, the tree can be expanded, revealing the next layer of its structure. This enables a view, including all details, if a user wants to investigate them.

A critical aim of the pattern debugger is to visualize the hierarchical order between observables, indicators, and incidents. To achieve this, for example, each incident is highlighted when the user is hovering over it with the cursor. Additionally, each underlying indicator and preceding observables are also highlighted to show the dependencies between them. This provides an overview of the hierarchical structure of the underlying correlation engine.

#### 4.4 Discussion of the prototype

As delimited before, the main requirements for the research prototype are to reduce the needed  $K_o^i$  and enable the *coll* between  $S_e$  and  $S_n$ . In the following, we discuss the degree to which it achieves to meet these requirements.

The most evident step, the reduction of the needed  $K_o^i$  for  $ext(K_d^i)$  is taken through the implementation of visual programming approaches within the Visual Pattern Builder. Thereby, users can create and analyze detection rules without being familiar with most SIEM systems' complex syntax. Thus, it is easier for them to contribute their  $k_d^i$  and externalize it into  $k^e$  (Eq. 7). Especially  $S_n$  are enabled to contribute their  $k_{d(nonSec)}^i$  to the definition of detection rules.

In addition to the rule creation, the complexity of rule debugging and testing was simplified. To avoid overburdening the user, only a very abstract view of

the recognized rules is displayed with a Live Event Chart. If a user wants to have a more detailed view, the Pattern Debugger can be used, which itself initially only displays the event names. If an even more detailed view is desired, these can be expanded further. Therefore, the prototype also simplifies *int*.

The points mentioned above already partly imply the fulfillment of these requirements since a reduction of the required knowledge enables  $S_n$  to participate in creating rules. This eases the exchange of knowledge in terms of collaboration, as defined in Equation 9. Thus, for example,  $S_n$  can adapt rules of  $S_e$  with their domain knowledge or vice versa. The prototype's architectural design also supports this knowledge conversion as patterns are stored centralized and made available through visual interfaces. Thus, the combination of *int* and *ext* are combined to enable collaboration as any user can access other users' externalized knowledge.

Although our prototype points in a promising direction for knowledge-based SA, its current state must be improved and thoroughly evaluated. We are aware of a few improvements to be made regarding a further simplification of the rule creation. Although the highly complex Esper EPL is more accessible with the current version, we aim to make rule creation even more straightforward. Additionally, the rule debugging functionality needs to be improved to give a more cohesive insight into the workings of the Pattern Matcher. Additional visual representation could be applied here. Also, the effects of reducing the necessary  $K_o^i$  have to be empirically measured to assess the prototype's full potential. An empirical user study furthermore needs to examine the effects of collaboration on detection rates and performance of the respective SA methods.

## 5 CONCLUSION

This paper presents a formal representation of knowledge in the field of Security Analytics. Building on a formalization, we establish a model of knowledge-based SA based on the Incident Detection Process. Therefore, the paper provides a sound basis for future research in the field of knowledge-based Security Analytics, as it brings the previously non-uniform and mostly verbal descriptions to a formalized and consistent level. Several segments of this model are identified to need more attention from academic research. To provide a first possible approach enabling externalization of domain knowledge and collaboration between security experts and security novices, we implement a research prototype system that uses

visual programming to reduce the amount of operational knowledge needed and make it easier to create incident signatures.

Although we present a prototype addressing some of the open challenges in knowledge-based SA, there is room for future research. First, it is necessary to further technologically support the collaboration between security novices and security experts. This paper presents the first approach to this, but the approach has to be improved together with potential users and finally evaluated throughout a user study. A second research direction is to integrate situational knowledge into SA better. Here, initial approaches already exist in the area of human-as-a-security-sensor, but these should be further developed.

## REFERENCES

- Ackoff, R. L. (1989). From data to wisdom. *Journal of Applied System Analysis*, (16):3–9.
- Ben-Asher, N. and Gonzalez, C. (2015). Effects of cyber security knowledge on attack detection. *Computers in Human Behavior*, 48:51–61.
- Chao, P.-Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education*, 95:202–215.
- Chen, M., Ebert, D., Hagen, H., Laramée, R. S., van Liere, R., Ma, K.-L., Ribarsky, W., Scheuermann, G., and Silver, D. (2009). Data, information, and knowledge in visualization. *IEEE Computer Graphics and Applications*, 1(29):12–19.
- Chen, T. M., Sanchez-Aarnoutse, J. C., and Buford, J. (2011). Petri net modeling of cyber-physical attacks on smart grid. *IEEE Transactions on Smart Grid*, 2(4):741–749.
- Davenport, T. H. and Prusak, L. (2000). *Working knowledge: How organizations manage what they know*. Harvard Business School Press, Boston, Mass.
- Dietz, M., Vielberth, M., and Pernul, G. (2020). Integrating digital twin security simulations in the security operations center. In *Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES)*, pages 1–9, New York, NY, USA. ACM.
- Eckhart, M. and Ekelhart, A. (2018). Towards security-aware virtual environments for digital twins. In *Proceedings of the 4th ACM Workshop on Cyber-Physical System Security - CPSS '18*, pages 61–72. ACM Press.
- Fayyad, U., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17(3):37.
- Federico, P., Wagner, M., Rind, A., Amor-Amorós, A., Miksch, S., and Aigner, W. (2017). 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)*.
- Heartfield, R. and Loukas, G. (2018). Detecting semantic social engineering attacks with the weakest link: Implementation and empirical evaluation of a human-as-a-security-sensor framework. *Computers & Security*, 76:101–127.
- Jaeger, L. (2018). Information security awareness: Literature review and integrative framework. In *Proceedings of the 51st Hawaii International Conference on System Sciences*. Hawaii International Conference on System Sciences.
- Loukas, G. (2015). *Cyber-Physical Attacks*. Butterworth-Heinemann.
- Mahmood, T. and Afzal, U. (2013). Security analytics: Big data analytics for cybersecurity: A review of trends, techniques and tools. In *2013 2nd National Conference on Information Assurance (NCIA)*, pages 129–134. IEEE.
- Menges, F. and Pernul, G. (2018). A comparative analysis of incident reporting formats. *Computers & Security*, 73:87–101.
- Nonaka, I. and Takeuchi, H. (1995). *The Knowledge Creating Company*. Oxford University Press.
- Polanyi, M. (2009). *The Tacit Dimension*. University of Chicago Press, Chicago.
- Ponsard, C. and Grandclaudon, J. (2020). Guidelines and tool support for building a cybersecurity awareness program for smes. In *Information Systems Security and Privacy*, volume 1221 of *Communications in Computer and Information Science*, pages 335–357. Springer, Cham.
- Sacha, D., Stoffel, A., Stoffel, F., Kwon, B. C., Ellis, G., and Keim, D. (2014). Knowledge generation model for visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1604–1613.
- Sáez-López, J.-M., Román-González, M., and Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school. *Computers & Education*, 97:129–141.
- Sallos, M. P., Garcia-Perez, A., Bedford, D., and Orlando, B. (2019). Strategy and organisational cybersecurity: a knowledge-problem perspective. *Journal of Intellectual Capital*, 20(4):581–597.
- Schneier, B. (2015). *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 15. edition.
- Schneier, B. (2018). *Click here to kill everybody: Security and survival in a hyper-connected world*. W.W. Norton & Company, New York, 1. edition.
- Thalmann, S. and Ilvonen, I. (2020). Why should we investigate knowledge risks incidents? - lessons from four cases. In Bui, T., editor, *Proceedings of the 53rd Hawaii International Conference on System Sciences*.
- Vasileiou, I. and Furnell, S. (2019). Personalising security education: Factors influencing individual awareness and compliance. In *Information Systems Security and Privacy*, volume 977 of *Communications in Computer and Information Science*, pages 189–200. Springer, Cham.
- Vielberth, M., Menges, F., and Pernul, G. (2019). Human-as-a-security-sensor for harvesting threat intelligence. *Cybersecurity*, 2(1).

- Wagner, M., Rind, A., Thür, N., and Aigner, W. (2017). A knowledge-assisted visual malware analysis system: Design, validation, and reflection of kamas. *Computers & Security*, 67:1–15.
- Wang, X., Jeong, D. H., Dou, W., Lee, S.-W., Ribarsky, W., and Chang, R. (2009). Defining and applying knowledge conversion processes to a visual analytics system. *Computers & Graphics*, 33(5):616–623.
- Zimmermann, V. and Renaud, K. (2019). Moving from a “human-as-problem” to a “human-as-solution” cybersecurity mindset. *International Journal of Human-Computer Studies*, 131:169–187.

