synERGY: Cross-correlation of operational and contextual data to timely detect and mitigate attacks to cyber-physical systems

Florian Skopik[a,1], Max Landauer, Markus Wurzenberger[a], Gernot Vormayr, Jelena Milosevic, Joachim Fabini[b], Wolfgang Prüggler[c], Oskar Kruschitz, Benjamin Widmann, Kevin Truckenthanner[d], Stefan Rass[e], Michael Simmer, Christoph Zauner[f]

[a]*AIT Austrian Institute of Technology, Center for Digital Safety and Security, Giefinggasse 4, 1210 Vienna, Austria*
[b]*TU Wien, Institute of Telecommunications, Gußhausstraße 25, 1040 Vienna, Austria*
[c]*MOOSMOAR Energies OG, Moosberg 10, 8960 Niederöblarn, Austria*
[d]*Huemer iT-Solution Ges.m.b.H., Leonard-Bernstein-Straße 10, 1220 Vienna, Austria*
[e]*Universität Klagenfurt, System Security Group, Universitätsstr. 65-67, A-9020 Klagenfurt, Austria*
[f]*LINZ STROM GAS WAERME GmbH fuer Energiedienstleistungen und Telekommunikation, Wiener Strasse 151, 4021 Linz, Austria*

Abstract

The degree of sophistication of modern cyber-attacks has increased in recent years, and in the future these attacks will more and more target cyber-physical systems (CPS). Unfortunately, today's security solutions that are used for enterprise information technology (IT) infrastructures are not sufficient to protect CPS, which have largely different properties, involve heterogeneous technologies, and have an architecture that is tailored to specific physical processes. The objective of the synERGY project was to develop new methods, tools and processes for cross-layer anomaly detection (AD) to enable the early discovery of both cyber- and physical-attacks with impact on CPS. To this end, synERGY developed novel machine learning approaches to understand a system's normal behaviour and detect consequences of security issues as deviations from the norm. The solution proposed by synERGY are flexibly adaptable to specific CPS layers, thus improving the detection capabilities. Moreover, synERGY interfaces with various organizational data sources, such as asset databases, configuration management, and risk data to facilitate the semi-automatic interpretation of detected anomalies. The synERGY approach was evaluated in a utility provider's environment. This paper reports on the general architecture and the specific pitfalls that needed to be solved, during the design, implementation and deployment of the synERGY system. We foresee this work to be of benefit for researchers and practitioners, who design and implement security systems that correlate massive data from computer logs, the network or organizational context sources, to timely detect cyber attacks.

Keywords: cyber security, anomaly detection, security information correlation, log and network data, cyber incident handling

1. Introduction

Common security solutions such as firewalls, intrusion detection systems (IDSs) and antivirus programs mainly apply blacklist approaches (Scarfone and Mell, 2007). These solutions digest signatures, created in advanced malware labs or compiled from community data collected in cloud-based malware detection systems, and periodically distributed to endpoints to enable the detection of known bad activities. While they reliably work for known attack attempts, they fail if new attack techniques are used or still unknown vulnerabilities or weaknesses are exploited. As a consequence, anomaly detection (Chandola et al., 2009) is required, which can discover even slight deviations of the system behavior, which might lead to traces of unknown attacks.

Unfortunately, most state-of-the art anomaly detection solutions are not easily applicable to CPS and operational technology (OT) (Mitchell and Chen, 2014), which fundamentally differ from enterprise IT networks in terms of complexity, size and widely distributed installations. Additionally, the few available OT security solutions are not built to work across different infrastructure layers (i.e., correlate information from OT and IT systems, as well as data from the network layer and log data from endpoints) and are thus mostly unsuitable to detect modern multi-stage cyber-attacks (Friedberg et al., 2015).

Another reason why today's anomaly detection systems are widely unsuitable for CPS is that CPS not only differ from enterprise IT networks, but also differ heavily among each other. CPS on the one hand have variable network infrastructures and on the other hand the integrated components are manifold. Hence, machine-learning approaches are required that do not depend on the peculiarities of a specific system for detecting attacks, but which adapt to different usage scenarios. Moreover CPS' operating characteristics differ from enterprise IT. Processes

*Corresponding author
Email addresses: florian.skopik@ait.ac.at (Florian Skopik), firstname.lastname@ait.ac.at (Max Landauer, Markus Wurzenberger), firstname.lastname@nt.tuwien.ac.at (Gernot Vormayr, Jelena Milosevic, Joachim Fabini), w.prueggler@mmenergies.at (Wolfgang Prüggler), firstname.lastname@huemer-it.com (Oskar Kruschitz, Benjamin Widmann, Kevin Truckenthanner), stefan.rass@aau.at (Stefan Rass), {m.simmer|ch.zauner}@linzag.at (Michael Simmer, Christoph Zauner)

are usually based on highly deterministic machine to machine communication, which allows more sensitive anomaly detection with lower system behavior deviation thresholds. Yet, existing anomaly detection approaches (Chandola et al., 2009; Aburomman and Reaz, 2017) barely take advantage of this.

Attackers can exploit vulnerabilities and weaknesses at different levels and in different areas of complex CPS (Humayed et al., 2017) as entry points for launching successful multi-step attacks. To counter these often advanced malicious activities, we argue that an appropriate composition of different detection approaches for individual infrastructure layers (WAN, LAN, field layer) improves the effectiveness of anomaly detection in CPS. These *cross-layer solutions* correlate multiple data streams of interest that are measured at different locations and OSI layers of the protocol stack. With the right combination and analysis methods, a cross-layer approach can increase the overall security situational awareness in CPS tremendously. Nevertheless, the selection of suitable data streams and observation parameters (e.g., observation points, time periods, granularity, and layers) remain challenging tasks. Besides detection quality, the required resources (time, budget, effort) for the detection is of importance.

We therefore propose an architecture and demonstrate a proof-of-concept implementation of a modern reactive security solution specifically designed for large-scale and complex CPS. The design criteria of this architecture are:

- It makes use of operational data sources of any type, particularly network data and endpoint data, and various areas of a CPS from the field level up to the enterprise level.
- It applies top-notch anomaly detection mechanisms, not just signature-based solutions, and uses machine learning to optimize the same.
- It utilizes cross-correlation techniques to increase the confidence in findings and to discover novel multi-step attacks that progress through an infrastructure.
- It facilitates the interpretation of discovered anomalies using contextual data from within an organization.

The synERGY project investigated how to design, develop and validate an architecture for an adaptive self-learning cross-layer anomaly detection system based on open standards, which can be vendor-independently applied to a wide range of CPS and can discover the tracks of a wide variety of modern cyber-attacks with limited human effort by applying cross-correlation techniques of numerous data streams.

Our main contribution is not about the core anomaly detection methods, but showing the various ways on how the integration of different components (even from different sources) could work to enable cross-correlation and a demonstration of the added value. The novelty lies in the complete system view, which we provide in much more detail than vendors would do for their commercial solutions. In particular, the contributions of the project (and this paper) are:

- An illustrative use case and reference architecture that enables cross-correlation to detect anomalies across system areas and layers of the technology stack.
- A discussion on potentially applied anomaly detection techniques for log data and network data.
- An implementation and Proof-of-Concept demonstrator at a utility provider's site.
- A critical discussion of the advantages and disadvantages of the cross-correlation approach, including a cost-benefit analysis.

Working on these challenges and research needs in the scope of a cooperative research project was important to avoid vendor-lock-in. Hence, particularly open source solutions and open standards were selected during development.

The remainder of the paper is structured as follows. Section 2 elaborates on background and related work. Section 3 describes an illustrative use case for cross-correlation. Section 4 shows the building blocks of the synERGY architecture. The anomaly detection on various types of data streams and applied methods are discussed in more detail in a designated separate Sect. 5. To prove its real-world applicability, we implemented the system and integrated it at a utility provider's site, as outlined in Sect. 6. We discuss the outcomes of our proof-of-concept study in Sect. 7 and conclude in Sect. 8.

2. Related Work

This section summarizes related work on the key aspects of the synERGY approach including architectures of distributed systems, data sources for cross-correlation analysis, intrusion detection in CPS, and incident response and decision making.

2.1. Distributed systems architectures

Distributed systems architectures (Tanenbaum and Van Steen, 2007) are conceptual models that design systems that consist of different software and hardware components which communicate via a network and coordinate their actions. Since several decades, distributed systems architectures have served as blueprints for numerous networks. In power grids, especially smart grids they define concepts for controlling several components (Overman and Sackman, 2010), forecasting energy demand (Hernández et al., 2013) and enable intelligent home energy management systems (Shakeri et al., 2017). Also modern applications such as smart living build on distributed systems architectures (Li and Jayaweera, 2014; Hu et al., 2016). CPS are distributed systems by definition and build on architectures that systematical model the deployment and connection of CPS to enable, for example, Industry 4.0-based manufacturing systems (Lee et al., 2015). Regarding cyber security in CPS, the risk architecture level of the Reference Architecture Model for Industry 4.0 (RAMI 4.0) model includes vulnerabilities and threat catalogs, as well as safety and security components (Ma et al., 2017). Settanni et al. (2018) describe how the Monitor-Analyze-Plan-Execute over Knowledge-based

(MAPE-K) (Arcaini et al., 2015) reference model can be utilized to apply anomaly detection to protect CPS. Due to the distributed nature of CPS, cyber attacks can target different layers including application, transport and control layer, which requires specific security architectures (Lu et al., 2015). Other approaches focus on distributed cyber security frameworks (Bolla et al., 2018). Such distributed security frameworks often connect various security mechanisms such as intrusion detection systems, vulnerability scanners and anti virus solutions. Therefore, often message queues are applied as event bus to collect information from multiple agents (Sadooghi et al., 2014). SynERGY follows a hybrid approach that provides an architecture that allows to utilize different systems to analyze data on host side and in the network. Furthermore, synERGY includes centralized services that correlate anomalies from different detection mechanisms, and enrich alarms with organizational context and cyber threat intelligence, as well as a graphical user interface that provides several dashboards, where the user on the one hand can review current security events and on the other hand can change configurations of all incorporated services. All these components communicate via the same event bus.

Combining open source solutions such as elastic's ELK stack[1] that allow to build data processing pipelines for collecting, storing, searching and visualizing data, and Apache's Kafka[2] or implementations of the open standard publish-subscribe network protocol MQTT[3] that serve as event bus, enable straightforward implementation of modern security architectures for distributed systems such as CPS.

2.2. Data sources for cross-correlation analysis

The synERGY architecture allows different data sources and multiple local agents to carry out intrusion detection, as well as to enable automated analysis and interpretation of security incidents. SynERGY collects data at host and network level, as well as information of different abstraction, comprising operational and contextual data and allows to apply different intrusion detection algorithms including decentralized and centralized approaches. Finally, cross-correlation analysis (Debar and Wespi, 2001) is used to combine all the information to detect attacks and adversaries in CPS.

Log data (Chuvakin et al., 2012; Yuan et al., 2012; Gerhards, 2009) is collectible at host level and is the smallest common denominator of data produced by all computer systems, services and network components. Log data allows to monitor a computer system's or network's current state and is therefore a valuable information source for attack detection in IT networks, including CPS. Well known services for collecting, distributing and accessing log data are syslog, journald and windows event log.

On *network* level, pcaps (Moustafa and Slay, 2015) and NetFlows (Hofstede et al., 2014) provide information on the communication that is going on within a monitored network. However, in practice, major challenges are associated with network traffic analysis including systems to record accurately timestamped network data (Micheel et al., 2001), the sheer amount of acquired network data (Ji et al., 2012), encrypted traffic (Velan et al., 2015), and compliance to legal requirements and privacy aspects (European Council, 2016).

Network traffic and log data are both records of *events* occurring within an organization's IT infrastructure (Kent and Souppaya, 2006). During the analysis of the collected data, an inference to the actual occurred event can be made. Recognized events do not have to be solely based on log and network traffic data, but can also contain information about existing vulnerabilities, attacks in the past or even *organizational context-data*, e.g. employee time tracking or vacation planning data (Gupta, 2012). The outcome of existing security solutions or anomaly-detection-systems can also help describing an event.

One main prerequisite for cross-layer anomaly detection is accurate time synchronization of all event sources (Fernandes et al., 2019; Löf and Nelson, 2014). Cross-layer correlation of distinct event types demands for events to be augmented by an accurate, globally synchronized timestamp when the event has been recorded by a sensor and when the matching log entry has entered the system.

2.3. Intrusion detecion in CPS

Due to the large amount of data produced in today's IT networks, intrusion detection has become a big data problem relying on machine learning and artificial intelligence (Wurzenberger et al., 2018a; Kubiak and Rass, 2018). The synERGY framework allwos for the application of different intrusion detection systems (IDS) (Sabahi and Movaghar, 2008; Garcia-Teodoro et al., 2009) on host and network level. Generally, three methods are used in IDS: (i) signature-based detection (SD), (ii) stateful protocol analysis (SPA) and (iii) anomaly detection (AD) (Liao et al., 2013; Scarfone and Mell, 2007). SynERGY aims at mitigating unknown threats and thus focuses on *AD* (behavior based) approaches (Chandola et al., 2009), which learn a baseline of normal system behavior, a so-called ground truth. Against this ground truth, all occurring events are compared to detect anomalous system behavior. A drawback of AD based IDS is the usually high false positive rate. Thus, synERGY aims at cross-correlating (Valeur et al., 2004) alarms from different anomaly detection systems to reduce false positives. Therefore, synERGY implements an hybrid approach for attack detection.

There exist plenty of works that discuss techniques and challenges for intrusion detection specifically focusing on CPS (Han et al., 2014; Mitchell and Chen, 2014; Humayed et al., 2017). Various of these IDS focus on anomaly detection and implement behavior-based detection algorithms (Kwon et al., 2015; Junejo and Goh, 2016). Liu et al. (2016) propose an unsupervised spatiotemporal graphical modeling approach to anomaly detection in distributed CPS. Harada et al. (2017) describe a log-based anomaly detection that uses a statistical method. Others use machine learning based methods to implement unsupervised and semi-supervised anomaly detection (Goldstein and Uchida, 2016) for CPS (Valdes et al., 2016; Inoue et al., 2017).

[1] https://www.elastic.co/what-is/elk-stack [last accessed 4/7/2020]

[2] https://kafka.apache.org/ [last accessed 4/7/2020]

[3] http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html [last accessed 4/7/2020]

Finally, there are approaches that apply artificial intelligence methods, including recurrent neural networks (Goh et al., 2017), bayesian networks (Krishnamurthy et al., 2014), autoencoders (Yuan and Jia, 2015) and deep learning (Wickramasinghe et al., 2018).

2.4. Incident response and descision making

The modular and open design of the synERGY architecture allows the application of many different IDS solutions. For handling alarms provided by different agents and to reduce false positives, synERGY applies a cross-correlation approach. Alert fusion is a heavily discussed and somewhat controversial topic. Besides simple fusion methods, such as binary AND, binary OR, majority voting, and weighted voting, a number of advanced approaches have been proposed and evaluated to perform alert correlation across 'IDS ensembles', some of them even incorporating contextual elements. For instance, Morin et al. (2009) proposes a logic-based model to support alert correlation; Elshoush and Osman (2011) focuses on fuzzy logic, soft computing and other AI techniques; while Bouzar-Benlabiod et al. (2010) incorporate security operator's knowledge and preferences, which is represented using the Qualitative Choice Logic (QCL). Another apporach suggests to use a decision-theoretic alert fusion technique based on the likelihood ratio test (LRT) (Gu et al., 2008). In synERGY, we apply numerical methods, for example, by computing a weighted average of all available risk scores (Alsubhi et al., 2008) or estimating detection confidences using statistical distributions (Roundy et al., 2017), since these are easily applicable for a wide range of anomaly detection components. Another approach, we consider is, to account for alerts as time-series and aggregate their deviations to obtain an anomaly score that describes dynamic system behavior (Landauer et al., 2018).

Furthermore, incident response and decision making includes interpretation of alarms using threat information collected by many different actors. Platforms, as described in (Settanni et al., 2017), provide the possibility to share open source intelligence (OSINT) from sources, including CERT lists (US-CERT and ICS-CERT), security-related mailing lists (e.g., Seclists.org), vulnerability databases (CVE, CVSS), threat intelligence feeds (hailataxii.com), threat assessment methods (Open-VAS , ESUKOM), antivirus reports and security bulletin (Microsoft Security Bulletin) (Skopik et al., 2016). Besides threat information, another valuable source to interpret alarms regarding insider threats and stolen user credentials is organizational context that includes information on access rights, employees' vacation days and sick leaves (Gupta, 2012).

Collecting and analyzing all these relevant data in one central system, can be done using security information and event management (SIEM) systems. A SIEM is an extended type of log management solution with basic features like log management, log analysis and graphical interpretations via dashboards and widgets, but also advanced features for event correlation, alerting, incident management, reporting, incident investigation, and more. A SIEM can collect data of different types - e.g. log data, network traffic data or contextual data - via various sources and communication channels - e.g. Syslog, Agents (software client to collect log data), Netflow or IP Flow Information Export (IPFIX). SIEMs can be used for historical incident analysis, and - even more important - for real time monitoring (Kent and Souppaya, 2006; Gupta, 2012). There exist various proprietary SIEM solutions. However, we defined purely commercial solutions (i.e., solutions where no free fork is available) out of scope of this work. An exception is the utilized Security Advisor, which however is based 100% on the ELK stack (Chhajed, 2015) and the features applied in synERGY could easily be re-implemented using ELK.

3. Use case for cross-correlation

Let us dig up the benefits of cross-correlation with help from a use case in the energy domain.

3.1. Use case outline

Reports of cyber-attacks on energy suppliers or other operators of critical infrastructure, including Stuxnet, Crashoverride, Black Energy or Petya (Hemsley et al., 2018), have increased in numbers recently. Regarding electric power distribution grids, a modern substation includes a number of safety-, monitoring- and control equipment. Our assumption for the use case of cross-correlation is that all technologically feasible security- and safety measures are implemented and functioning correctly. However, since the life cycle of industrial components in the energy industry is rather long compared to standard IT, the existence of legacy devices having extra protection requirements is quite common. Due to these conditions, the application of anomaly detection is a promising means to further protect such systems. When applying advanced anomaly detection systems, the primary protection goals in the field of industrial security, availability and integrity, are of utmost importance.

In particular, the real time properties of industrial systems must not be restricted. This is a significant difference to the classic office IT world, where confidentiality is usually a top priority (Shaaban et al., 2018). A vital concept in the area of industrial security is *defense-in-depth* (Rass and Zhu, 2016), which is based on the recognition that protecting against cyber attacks to industrial installations, such as the power grid in our use case, involves all stakeholders such as operator, integrators and manufacturers. In this *shell model*, the attacker must first overcome several protections, each representing a line of defense, to advance to the next level.

In addition to building a defense-in-depth concept, correlating detected anomalies from the different layers of the shell model is vital to detect well-hidden attackers earlier and increase the quality of the alerts, i.e. decrease the false positive rate. From the perspective of overall security, correlating detected network traffic anomalies with physical security factors such as access alerts, work orders, etc. has great potential. The synERGY use case therefore accounts for these factors.

3.2. Use case infrastructure

According to the IEC 62443 series of standards, the network structure of a modern distribution system operator is structured

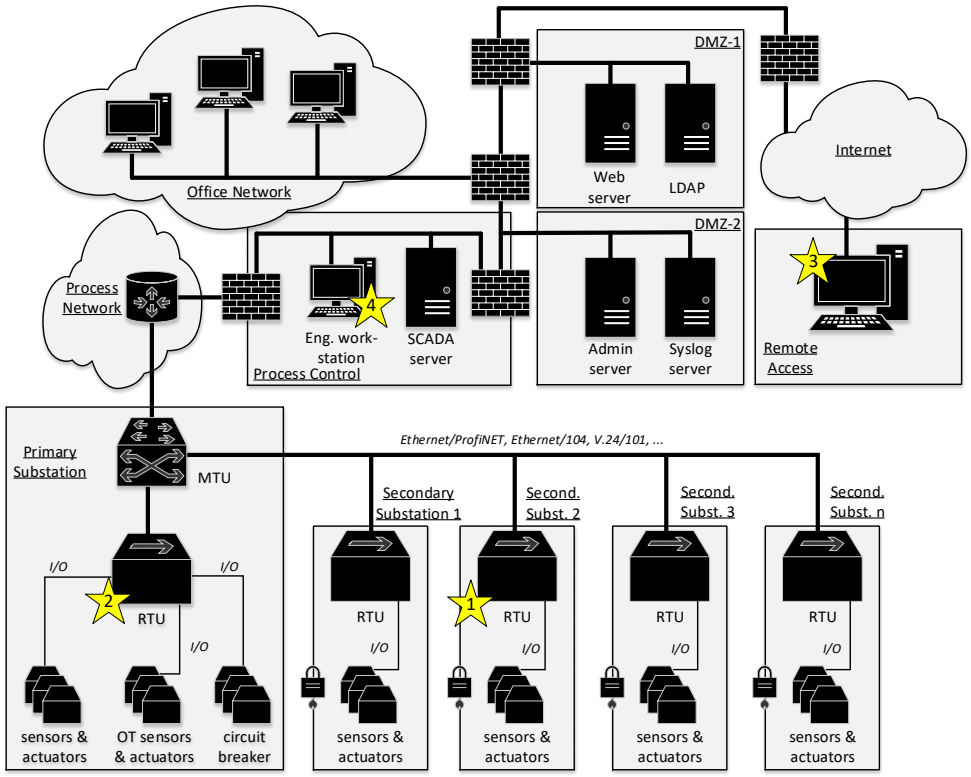

Figure 1: A typical infrastructure of a utility provider. The yellow stars mark weak spots on the attack surface.

as given in Fig. 1. This includes a system architecture with differently protected zones and transition points (conduits) (Shaaban et al., 2018). Specifically, the network structure is divided into five zones and interconnected by defined conduits: (i) Zone 1: actuators and sensors, (ii) Zone 2: process network, (iii) Zone 3: supervisory control and data acquisition (SCADA) system, (iv) Zone 4: office network, and (v) Zone 5: Internet.

In the SCADA system, substations are connected via Remote Terminal Units (RTUs), multiple RTUs usually communicate with a Master Terminal Unit (MTU). As physical media fiber optic cables, as well as copper lines or radio links are used. Based on this, different communication media are used and in particular network technology such as Ethernet. The transmission protocol between the SCADA or control systems and the substations is the standard IEC 60870-5-104 (short: protocol 104), which is based on TCP/IP and is applied for basic telecontrol tasks. The protocol 104 transmits data in plain text without cryptographic measures. The message can only be interpreted if it is known which address belongs to which actuators/sensors. Some of the substations also use the IEC 60870-5-101 protocol, where serial links are typically used, as well as ProfiBUS or ProfiNET. In this environment multiple standard processes exist. In this paper, we specifically focus on the maintenance processes, since they represent exceptional situations, which are particularly hard to monitor:

On-Site Maintenance process: The entrances to the substations are equipped with door contacts and when the door is opened an alarm is triggered in the network control center. Therefore, the authorized persons are obliged to log in to the network control center before they start the maintenance work, as well as to log out when leaving the substation. If no person has logged on in the event of an alarm, a verification process becomes active. The maintenance itself is announced prior to implementation at the grid control center and must be released by the latter in the defined maintenance window.

Remote maintenance process: Usually, the user can access the office network after a successful two-factor authentication. From the office network, authorized persons have access to a server, which regulates the access rights to the individual MTUs and RTUs.

3.3. Types of attacks

Unauthorized access to the process network or systems is one of the biggest threats to the overall system and forms the basis for a variety of threat scenarios. Such access can occur, either physically or logically over the network, to virtually every component and every domain of the network. Particularly critical is unauthorized access to components in the field, as an attacker may face only little or no physical access protection there. Access allows the attacker to copy, listen, or modify data that may be transmitted and use it for his/her own purposes. This use case specifically addresses the following four attacks (also marked as stars in Fig. 1):

1. Attacker physically breaks into a secondary substation station and gets access to an RTU.
2. Attacker physically breaks into a primary substation and gets access to an MTU.

3. Attacker gets access to the network via remote maintenance access.
4. Attacker gets access to the network via a compromised device, e.g., an engineering substation or maintenance notebook.

3.4. Impact of attacks

Attacks may have various levels of impact, depending on where they are carried out and for what purpose. It is usually a best practice to avoid an overly crisp quantification in monetary units, or fractions of loss on market share, customers, or other numeric means. A practically more viable method is using an even number of *impact categories* (even numbers avoid a "non-informative middle category" that a scale with an odd number of categories would have), whose detailed definition is up to the practical application domain (Rass, 2017; Wachter et al., 2017; Kammerstetter et al., 2014).

3.4.1. Disruption of communication

The disruption or the blocking of communication between different network components represents a common attack vector. The goal may be to affect the availability of components or services, i.e., a denial-of-service (DoS) attack. At the same time, the intention of an attacker could be to prevent the transmission of measurements or commands sent to a device by stopping communication. This could cause wrong control decisions or generally lead to instabilities of the grid due to a lack of information, or that certain actions, such as switching commands, are not performed accordingly.

3.4.2. Eavesdropping of communication

Attacks of this type attempt to listen to, but not change, communication data. This can be done completely passively. Attacks of this kind are not necessarily aimed only at plain text communication. For example, certain patterns of encrypted communication may allow conclusions about the information transmitted. A possible target of an attacker may be to obtain information about the network and the components and technologies used therein. Information obtained in this way could subsequently be used for targeted attacks. We anticipate the following types of attacks: (i) Packet sniffing attacks: The attacker uses a wire tap or a monitoring session on the switch to monitor the traffic on this port. (ii) ARP cache poisoning or ARP poison routing: ARP packets are inserted into the network to change the routing or to output the traffic on all switch ports.

3.4.3. Active manipulation of communication

This represents a particularly severe threat if the communication data can be actively changed by an attacker or even new data introduced (person-in-the-middle attack). Measured values could be manipulated by an attacker or outdated values reintroduced. The following types of attacks need consideration: (i) Packet injection: The attacker inserts additional packets within an existing network connection. (ii) Replay attacks: A valid data transfer will be recorded by the attacker and retransmitted at a later point in time. (iii) Data manipulation: The attacker modifies the contents of packets during transmission. (iv) IP spoofing: The attacker brings packets into the network with fake sender IP addresses. (v) IP and TCP protocol attacks against network stacks of terminals or firewalls on the route: e.g. fragmentation attacks, Ping of Death, etc. (vi) ARP spoofing: The attacker acquires the MAC address of a legitimate device and uses this MAC address in a spoofed device.

The manipulation of measured values or other status information can lead to instability of the distribution grid. For instance, if demand-response systems work with fake data, it may lead to over or under supply of the network with energy (Skopik et al., 2012). Sending fake commands or re-importing previously recorded commands could allow an attacker to trigger unauthorized actions on components. At the same time, valid and authorized commands could be suppressed by an attacker. If in addition a previously recorded acknowledgment message of the actual recipient is re-played, the non-execution of an action could be hidden for extended time periods.

Likewise, an attacker could use infiltrated or manipulated data to alter the configuration or firmware of components in order to influence their behavior or bring them under their control. Another motivation to introduce manipulated data in the communication between the individual components may be to perform DoS attacks. Especially considering that many network components are embedded devices with limited resources, DoS attacks using data flooding are realistic scenarios. Regardless of the exact purpose of the attack, any manipulation of communications within the process network will in any case mean the loss of reliability and traceability of actions and events.

3.4.4. Manipulation of hardware or software

If an attacker gains access to development or administration systems, this can allow the execution of targeted actions on various components along the entire value chain. Compromised centralized systems could also allow an attacker to deliver compromised firmware to the components of the process network. In addition to widespread attacks on the SCADA system, a threat scenario is that individual components are directly manipulated. This could be achieved by changing or replacing the firmware transferred to a device. Particularly vulnerable to attacks of this kind are components in the substations, since these are easy to access for an attacker. In addition to manipulating the logic of the components, another threat is changing data processed or used by the devices, including metrics, certificates or other authentication data.

3.4.5. Malware

Similar to other computer networks, the process network in our use case is threatened by malware. Malware-infected systems could allow an attacker to control other systems or tamper with sensitive data. A special case thereof are *backdoors*, which allow stealthy access to a system. On the one hand, such access can be deliberately implemented by the manufacturer as maintenance access, or can be created by an attacker to gain access to a compromised system at a later time.

Table 1: Traces of attacks on the hosts (logs), in the network, or in the physical environment of a CPS.

traces/sources	**further description and remarks**
failed logins	series of failed login attempts (especially those where a successful login follows immediately)
permission changes	changes of r/w permissions, creation of a new administrator account
configuration changes	configuration changes (i.e., date, user who made the change and what has been changed) on switches, particularly the creation of a new VLAN or other security-related configurations, including changes to cryptographic parameters (updates of keys, roles and access permissions)
implausible data	comparison of sensor values with historical data may lead to the detection of deviations
time base settings	deviations of the timestamps of log data of the RTUs and the SCADA server
user authentication	registration of a successful authentication of a default or administrator user, deviating parameters of the authentication, e.g. used authentication method (password, SSH), SSH key fingerprint, protocol settings used (hmac-sha1, aes, ...), session settings (TERM variable, ...)
authorization	execution of commands that according to the user role concept are not allowed
logout/logoff	The logoff time should be logged in the same way to detect 'silent' acceptance of authorized connections for long-term use.
altered or deleted log entries	manipulations on the network devices (router, switch, firewall) or host systems
device boot up	visible in the log entries due to numerous unique startup events
traffic statistics	anomalous traffic patterns, captured with tools, such as Flowmon[4], for instance, communication from one MTU to another MTU
comparison of traffic profiles	time windows-based comparison of current traffic volumes (netflows) with historic data; classification of flows based on number of packages and sizes. Note that traffic profiles of similar substations can be compared to one another, if they possess similar sensors and serve similar actuators
broadcast	broadcast storm directly on the switch
device authentication:	failed authentication attempts of fake devices in the process network through the NAC of the switches (MAC-based and where technically possible over 802.1X)
ARP spoofing	independently detected by the switch and relayed as alarms
loss of communication	An interface, which goes down, is quite common during normal operations, but together with other anomalies a good indicator that something is odd.
overload of a communication interface	e.g., by a DoS attack. This can easily be detected if a station is not reachable (failure messages in the SCADA system).
changes in the protocol 104	injection of custom non-standard packages used to manipulate stations
data from protocols other than 104	other ports or package formats are used
newly added or missing devices	Devices with sender IP address are visible (either via ARP requests or DHCP requests) in the network which can not be found in the asset DB.
network Port goes down	All unnecessary ports are disabled by default; however, ports often also go down temporarily in normal operation.
ethernet connection parameters change	e.g. "Eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: None". Since embedded devices in particular often do not use the maximum power settings, the temporary attachment of a notebook instead of the original device can be detected, even if the MAC has been correctly spoofed.

3.4.6. Authentication threats

Inadequate or insufficiently implemented authentication measures also constitute a major threat. Many components can be shared by several people, such as, maintenance personnel could share a common user account. This affects the traceability and clear assignment of actions as well as the security in general, since authentication data is shared between many different users. In addition, the same authentication data may be used on a large number of devices. The likelihood and impact of compromising authentication data is greatly increased by these circumstances.

3.5. Detectable anomalies across components and layers

Multiple types of anomalies are already detectable with state of the art technologies. Some of the relevant anomalies for the system given in Fig. 1 are listed in Table 1. For discovering traces of intrusions and detecting single anomalies as described in Table 1, various often specialized solutions exist (Aburomman and Reaz, 2017; Sabahi and Movaghar, 2008; Mitchell and Chen, 2014; Fernandes et al., 2019). The focus of synERGY however is on the correlation of such 'simple' anomalies across layers and components to detect complex (potentially multi-stage) attacks more precisely on a higher level.

3.6. Contextual sources for anomaly interpretation

For the interpretation of an attack, the following data sources would provide additional support for the interpretation of discovered anomalies. This interpretation helps to categorize anomalies and estimate their criticality and impact:

- **Risk management:** An IT and information risk management system according to ISO 27001, where potential risks to certain assets are systematically assessed, is beneficial. This information can aid in both interpretation and prioritization of anomalies.
- **Vulnerability Management:** Systematically recorded vulnerabilities and information about asset weaknesses help to prioritize anomalies. In other words, if a component is known to be vulnerable and it starts to show anomalous behavior, the investigation of root causes should be of priority.
- **Asset / configuration management:** The assets are documented in an IT service management system in the form of an asset list, where the changes to the assets are also documented. A comparison of asset data to monitoring data (e.g., devices, IP addresses in use etc.) reveals any unauthorized changes to the system.
- **Maintenance tasks:** A link between anomalies and working hours or maintenance tasks makes sense to detect if an anomaly could be caused by a potential attacker or whether it was triggered by a maintenance activity.
- **Employee hours:** A correlation with the actual working hours of employees would provide an additional plausibility check, whether the origin of an anomaly is an employee performing a work assignment, or possibly a potential attacker.
- **Manipulation sensors:** Another important external source are physical manipulation sensors that register certain events.

These include, for example, access sensors or door contacts. If there is no known maintenance activity scheduled, the trigger is most likely an intruder.

- **Monitoring system:** Devices affected by announced maintenance are declared in the monitoring system before maintenance to prevent the generation of false positive alerts.

4. The synERGY architecture and its building blocks

The overall architecture as shown in Figure 2 consists of several separate components, each of which is responsible for a part of the cross-layer intrusion detection. These components consist of the CPS to be monitored, the agents that collect the required data, the central data broker, a filter or normalization component, the AD systems, a centralized data repository, a SIEM system, external database connections, and a reporting system.

In addition to the components, Figure 2 shows all interfaces between these components, as well. The lines representing individual interfaces are augmented by arrows that represent the schematic flow of the data (regardless of whether pull or push-based protocols are used) - more on the workflow in Sect. 4.1. A description of the main components can be found in Section 4.2. Section 4.3 presents an overview on interface requirements, technologies and concepts used in synERGY.

4.1. synERGY workflow

In short, the principal synERGY workflow starts at the bottom left of Fig. 4.2. Here a CPS is monitored and all kinds of operational data (logs, network flows, etc.) are fetched from the system (via sensors, proxies, agents – whatever is appropriate for the given CPS) and fed via data diodes to the synERGY data broker (essentially a high performance message queue). The only exception are raw network data which are directly fed into the network AD component due to performance reasons. All other AD systems read the fetched data from the data broker and further process it independently. Results are stored in a central data store, together with a filtered and normalized version of the raw operational data. Eventually, the SIEM solution on top reads in the pre-processed raw data from this store and enriches these with the results of the anomaly detection systems. The SIEM further access contextual data, including a CMDB, data from physical access systems and external threat data, and utilizes these sources to support the cross-correlation process. Cross-correlation merges the single AD results into one overall anomaly score. The concrete mechanisms are explained in Sect. 5.3 but involves careful weighting of each input, a rule-based or arithmetic calculation and interpretation with the help of contextual elements. A dashboard visualizes the final output and a reporting system enables the escalation of events in case of results that exceed an acceptable anomaly threshold.

Notice, in Fig. 2 solid arrows represent flowing data, dashed arrows reconfiguration commands, e.g. to reconfigure the detection capabilities of anomaly detection components ot to whitelist events. Dash-dotted lines originating from the CPS data sources reflect raw unfiltered and unprocessed data.

4.2. Architectural components

The subsequent subsections describe the tasks of the individual components and requirements on their interfaces. Along this description, we will repeatedly refer to input and output interfaces, processing and delivering *data items* and *events*, labeled as (1)...(9) in Fig. 2.

4.2.1. CPS

The CPS in Figure 2 shows the system to be monitored. It contains networked sensors and actuators. Networking can use different technologies, such as Powerline or Ethernet. This network can be geographically distributed or concentrated in one location. These systems are usually implemented in a hierarchical structure with strict communication requirements resulting from limitations of the technologies used in CPS on the one hand and from security reasons (segmentation) on the other.

Anomalies in the CPS can potentially be detected either from network data or from log data of the individual components located in the CPS. Mechanisms for retrieving these data depend on the components and networks that build up the CPS, the location of installed sensors and their supported protocols. Agents, which are described in the following subsections, are responsible for data retrieval, processing and forwarding to the relevant entities.

4.2.2. Network agents

The network agents provide the network data that is needed for the network AD. These data can be acquired via port mirroring, network tap, or directly at terminal devices. The collected network data is processed and forwarded to the network AD via the network data interface. In addition to converting the data to the format required by the interface, the agent must add the preconfigured asset identification. Network agents implement one single interface: **Output:** Network data (1).

4.2.3. Log data agents

A log data agent is a sensor that collects data in packet form (e.g., SNMP). All fields provided by the source must be converted by the agent to a well-defined format and additionally augmented with the preconfigured asset identification. Log data agents implement one single interface: **Output:** Log data (3).

4.2.4. Log text agents

In contrast to the log data agent data, the data of the log text agent consists purely of text. Examples include, but are not limited to reading log files, a Windows event channel, or syslog entries. Therefore, only this text line can be forwarded as a single field.

Log text sources can supply these text lines in different encodings, which can also differ line-by-line for one single source. A re-encoding to one common synERGY format is mandatory but can lose relevant information. This is why log text agents transmit two copies of the log line: First the original text line, which must be converted to a safe format for transmission, and second a newly encoded text. The original line format is required for the automated analysis, i.e., anomaly detection. The

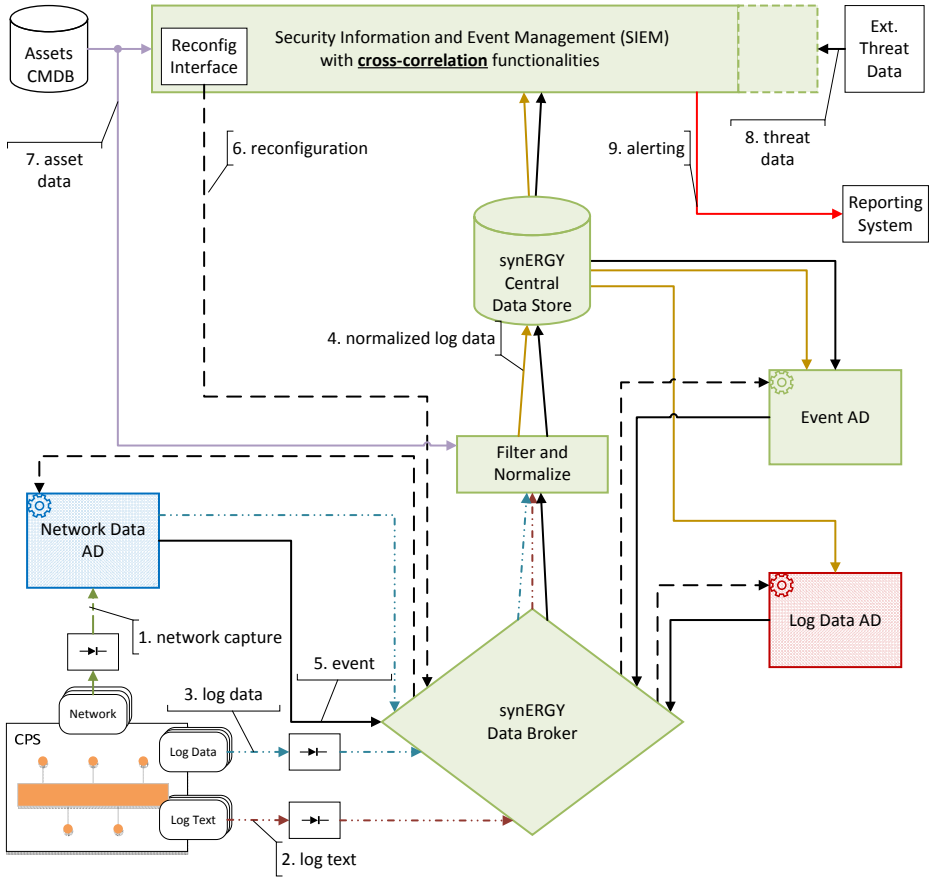

Figure 2: Architectural overview of the synERGY system.

second, converted format supports text search within the log lines and simplifies the SIEM representation. Log text agents implement one single interface: **Output:** Log text (2).

4.2.5. *Data diode*

A data diode is a hardware component that unconditionally restricts the flow of network data to one single direction. This feature recommends data diodes as mechanism of choice whenever critical networked components or subsystems must be shielded from outside access. It is important to note that this physical restriction prevents protocol feedback and in particular on-demand retransmission at all layers. Data diodes do not allow retransmission of lost data records on detecting losses, neither at transport layer (data diodes block the TCP protocol), nor at application layer.

In the synERGY architecture, data diodes decouple agents from systems that are located higher in the hierarchy. This means that sensors and agents are physically protected against, e.g., access by potentially compromised management software. Log data can flow exclusively from agents to systems located upwards in the hierarchy but never vice-versa. In addition, data diodes allow data to be collected within one network and then forwarded to another network. This enables implementation of the same network hierarchy in the monitoring system as in the CPS system.

4.2.6. *Data broker*

Key to the architecture in Figure 2 is the central communications component, the broker. This component performs message forwarding and buffering between the individual components. The synERGY implementation and all following discussions rely on Apache Kafka[5] as broker but support its replacement by other broker platforms having similar functionality.

Apache Kafka is a distributed, partitioning and replicating service that can be used for handling any kind of data stream. Similar streaming platforms have three key capabilities: (1) Publish and subscribe mechanism on streams of records, similar to a message queue or messaging system, (2) storage of streams in a fault tolerant and persistent way, and (3) processing of streams in their initial order. Kafka is used in two large classes of applications. First, real-time streaming data pipelines that reliably convey data between distributed systems or applications and second, real-time streaming applications that convert between distinct stream formats and/or process them. On this behalf, Kafka is implemented as a cluster of servers that can be distributed over several data centers. The Kafka cluster bases on the concept of *topics*, denoting logical instances to which messages belonging to specific categories can be sent and from which messages can be read.

[5]https://kafka.apache.org/ [last accessed on 12/23/2019]

Any Kafka data entry (record) consists of a key, a value and a timestamp. All messages exchanged between synERGY components use the Java Script Object Notation (JSON) format (Bray (2017)). JSON itself lacks the schema concept known from other formats like XML, thus synERGY makes use of JSON-Schema for validating messages and for verifying the compatibility of components with the synERGY architecture (see Sect. 4.3).

4.2.7. Filtering & normalization

Received log data must be normalized before storing it to support further processing. As part of this process, relevant information is extracted from the individual log lines and converted to a common format. On this behalf templates (patterns) must be defined for the used log formats. Such template defines the syntax of data within a given log format. Pattern matching is used to compare the incoming log lines with the stored formats and to extract relevant information. On the one hand, this normalization supports an optimized correlation of the data in subsequent analysis stages and, on the other hand, enables a better representation of the data in the graphical user interface of the SIEM.

The main purpose of filtering is to extract relevant information such as IP addresses, device names, date and time stamps, user names, status codes, error codes, actions performed, etc. from the log entries. The filtering module assigns a local timestamp to any received log data and event. This timestamp establishes an ordering of all incoming data that originated from potentially distinct sources. The main benefit of the ingress timestamp is that, as opposed to the timestamps stored in log entries by agents, it does not depend on time synchronization between the agents. In addition to these tasks, the filter and normalization component must convert the synERGY internal asset identification numbers into operator-specific asset identifiers using the asset database. Interfaces: **Input:** log text (2), log data (3), event (5), asset data (7), **Output:** Normalized log data (4), event (5).

4.2.8. synERGY central data store

The data storage serves as a central database for all collected log and event data. In addition, the data file serves as a generator for unique identification numbers for events, log text, and log data. All data stored in the data repository is accessible via the SIEM for further analysis and visualization. This includes data search and sequential retrieval. Furthermore the Event AD and the Log Data AD require the possibility to query all data in sequential order as soon as they have been stored in the database. Interfaces: **Input:** Normalized log data (4), event (5), **Output:** Normalized log data (4), event (5)

4.2.9. Network anomaly detection

Main task of the network AD module is to detect anomalies within the network data and to evaluate them. Network data is received by network agents using the network data interface and forwarded to the data repository via the event interface. Another function of this component is to process and filter the incoming network data. The processed data is forwarded to the data repository and can be used by the SIEM to display the context or detailed time sequences of detected anomalies. Interfaces: **Input:** network data (1), reconfiguration (6). **Output:** Log data (3), event (5).

4.2.10. Event anomaly detection

For event-based anomaly detection, known or hypothetical scenarios that describe a security incident must be defined in advance. These predefined scenarios are saved as patterns. During the processing and analysis of incoming log data, these are always compared with the predefined patterns. If a match occurs, an event is generated. A simple example would be a brute force attack. The pattern defined on this purpose could be, e.g., a series of 100 or more subsequent 'failed login' attempts, followed by a 'login success' log line, which suggests that the attack was successful. If such a pattern is detected in the received log data, an event is generated and an alarm is triggered. Interfaces: **Input:** normalized log data (4), event (5), reconfiguration (6), **Output:** Event (5).

4.2.11. Log anomaly detection

The log data based anomaly detection processes streamed log data without the need of log data storage, caching or access to reference data. This way it is also avoided that sensitive log data has to be securely stored in another location. In contrast to most systems that work with blacklisting approaches, the analysis is carried out with learning whitelisting instead, to recognize disturbances or attacks that are not yet described by known blacklisting patterns. For this purpose, the anomaly detection consists of two sub-modules: (1) A module for feature extraction: this extracts all data fields from log data as text, converts them into corresponding data types for subsequent processing (e.g., the text representation of a date into a date value) and makes them available to the detection module. (2) The detection module shall try to detect the normal state of the system from extracted values or value combinations and report deviating entries.

If a deviation is detected, a human readable message is generated, which describes the detailed cause of the anomaly. Details include (a) the detection module that detected the anomaly, e.g. normal distribution change detector, (b) related detector parameters, e.g. mean values and standard deviations, (c) amount of deviation, e.g. last set of values was within some confidence interval, and (d) reference some or all log lines, which had caused this deviation, e.g. 'Apache response time was 12 ms'. Interfaces: **Input:** Normalized log data (4), reconfiguration (6), **Output:** Event (5).

4.2.12. SIEM

The SIEM system is supposed to act as a central interface between the overall system and the user. It offers a graphical web interface for the detailed representation and analysis of all existing data in the data repository. The user has the possibility to search the data by means of a *full text search* or can display different *graphics*. Through the integrated *workflow management* it is possible to assign detected anomalies to responsible persons and to track the current status (e.g. new, in progress,

fixed). The *integrated asset management* shows an overview of all existing systems in the network and offers the possibility to configure installed agents on these systems. The SIEM also offers integrated *vulnerability management*. Vulnerabilities of vulnerability scanners (e.g. Nessus) can be transferred to facilitate analysis of the system and show the current security status of the network. Interfaces: **Input:** normalized log data (4), event (5), asset data (7), threat data (8). **Output:** Reconfiguration (6), alarm message (9).

4.2.13. Asset database

An asset database is accessible via the SIEM to integrate company-specific context data into the system. The most relevant information therein is which devices exist in the entire network and how they are configured. Among others, this can be used to check whether a device was unexpectedly added to or removed from the network. In addition, installed software versions and user access rights can be checked. Interfaces: **Output:** Asset data (7)

4.2.14. Threat data

Data on past security incidents is made available in a threat database. Here, for example, external IP addresses can be queried to check whether they have already been referenced in connection with a past security incident. Such databases are usually offered as online services, which can be accessed as required. However, since synERGY processes a substantial amount of data, this would cause a lot of traffic to the outside and the database operator could also create profiles for remote search queries. Therefore, a separate threat database was built and integrated into the system. This database is filled at regular intervals with data from various online services. Thus the actual enrichment of the log data is maintained as part of synERGY.

A further important point of the threat data concerns known and generally existing weak points of the used devices in the enterprise environment. If the system knows about a vulnerability (which, e.g., cannot be fixed for various reasons), the system can explicitly monitor special actions in connection with this vulnerability to ensure an early detection of the exploitation of this vulnerability. Interfaces: **Output:** Threat data (8)

4.2.15. Reporting system

Dedicated reporting systems must be integrated into synERGY to notify users promptly of any anomalies that have occurred or attacks that have been detected. Use cases of the reporting system are: email sent to a distribution list for direct mail notification of users, alerting over a SMS gateway, alerting via the SIEM to integrate messages into normal alerting and escalation procedures. Interfaces: **Input:** Alarm message (9)

4.3. Interfaces

Interfaces in synERGY that involve the data broker rely on Apache Kafka as outlined in Sect. 4.2.6.

Since sensors in synERGY are potentially exposed to physical access by adversaries, all receiving interfaces, and in particular the ones to sensors, must accurately verify the correctness and plausibility of received data. This includes verification of syntax, semantics, and data origin authentication. With respect to syntax, receivers must verify the correct sequence, count, length, and coding of data records within received messages. On this behalf, the format of JSON messages on any interface must unconditionally match an interface-specific, pre-defined JSON-Schema, otherwise they are discarded. The schemes for all interfaces reference a common base schema to avoid the use of redundant or incompatible definitions for identical data elements on distinct interfaces. In terms of semantics, receivers are required to verify consistency of the message content in the value domain, including compliance with data value range limits. Finally, data origin authentication is used to safeguard that cryptographically signed data records have not been modified by adversaries on the network path between the sensor and the receiver.

One notable exception to the JSON-based interfaces is the network capture interface (1) between the CPS network and the network anomaly detection. Because of the large amount of binary data the JSON format is not appropriate. Therefore the network capture interface relies on the IPFIX format (Claise et al., 2013) which supports the transfer of packet data, or, alternatively, of aggregated flow data. Moreover, IPFIX supports the transfer of additional synERGY-specific fields like, e.g., asset ID, and several transport settings, e.g., Transport Layer Security (TLS) encryption.

The selected protocols and implemented mechanisms ensure that the data transfer and storage within the synERGY architecture is at all times protected from unauthorized access or misuse.

5. Anomaly detection and interpretation

In terms of anomaly detection, synERGY focuses on (1) network data (Sect. 5.1) and (2) log data (Sect. 5.2). Results of independently discovered anomalies are correlated and interpreted with the help of contextual elements (Sect. 5.3).

5.1. Anomaly detection on network data

The designed network traffic analyser focuses on learning representations of normal network traffic, and based on this knowledge is able to identify when unusual instances, thus anomalies, happen. The analysis takes three steps: data preprocessing, building the model using autoencoders and evaluation of the trained model in practical scenarios.

5.1.1. Data preprocessing

The anomaly detection system uses network flows as the basis for analysis. A network flow consists of several network packets that have the same protocol type, source address, destination address, source port and destination port. Flows are terminated either after a timeout, or as soon as a packet signals a connection end. However in our scenario, since the SCADA protocol uses endlessly long transmission control protocol (TCP) connections, we use network sub-flows, which are based on

flows, but are terminated as soon as a packet with useful content is sent in the opposite direction to the current one. This corresponds to the behavior of the request-response model, in which one communication user asks a question and the other participant answers.

In the second step of data preprocessing, we extract features from analyzed flows and remove all non-numerical ones. This means that from the analysed features we drop information related to source and destination IP addresses, so as about source and destination media access control (MAC) addresses. Such selection is performed in order to force the trained model to generalise better and to learn rules that can be applied independently from these parameters. After analysis is done in such a way, we are left with 20 numerical features to work with in further analysis. It is important to point out that extracted features do not describe the packet content, which would not be possible with encrypted communication anyway, but rather provide information about packet properties and distributions of the included packet properties (e.g., statistics on number of incoming packets, lengths of packets, statistics on inter packet arrival times, etc).

The next step comprises data normalization, which is required to facilitate the analysis of data and avoid misleading importance, which some features would have simply by having larger range of value.

5.1.2. Building the model using autoencoders

To make the designed system suitable to be deployed in practice, we choose the main machine-learning component to be based on autoencoders (Goodfellow et al., 2016). Main reasons for such a decision are as follows:

1. Autoencoders are unsupervised methods that do not require labelled data which is in line with our detection environment where most of the observed traffic comes from normal traffic behavior.
2. Autoencoders can learn which features are the most representative for the representation of the normal traffic, thus automate the feature selection procedure and support the creation and extraction of sophisticated detection patterns.
3. Autoencoders learn to represent the input data using less dimensions. They are neural networks, but similar in principle to the dimensionality reduction techniques like Principal Component Analysis (Jolliffe, 2011) and Independent Component Analysis (Hyvärinen et al., 2001) in sense that they map data from higher dimensions into lower dimensions. However, unlike the other two reduction techniques, autoencoders are non-linear which allows them to capture and interpret more complex relations between observed data.
4. Depending on the number of hidden layers they can be very complex and may thus be unsuitable for constrained environments, or they can be made relatively simple and fast towards manageable requirements in terms of computation complexity: time, speed, memory. The complexity can herein be tuned.
5. Autoencoders have been successfully applied for anomaly detection (Zhou and Paffenroth, 2017; Sakurada and Yairi, 2014), so as for network traffic analysis and intrusion detection (Javaid et al., 2016; Meidan et al., 2018).

Data used in training and testing comes from benign traffic only, with the goal that an autoencoder focuses on learning to encode and reconstruct benign traffic as accurately as possible. For the training, some parameters need to be set: the number of encoding layers (the number of nodes), the number of encoding dimensions, the batch size, a loss function, and a metric. The number of encoding layers controls how much information the autoencoder can infer about the data. However, too many layers increase complexity of a solution and both training and testing take longer. For this reason, the selection of a suitable *number of layers* is a tradeoff between complexity and detection performance. In our experiments we looked into both shallow and deep autoencoders and experimentally found that a deep autoencoder with five hidden layers and following number of nodes: 15, 10, 10, 10, and 15 is the most suitable according to the used mean square error metric to capture existing relations among our analyzed input data.

The *activation function* is a parameter that determines the value that the node will send forward to the next layer based on weighted sum of its received inputs. In our experiments we use a rectified linear unit activation function in all layers except the output layer. This is the most commonly used activation function in neural networks (LeCun et al., 2015), since it is at the same time effective and simple.

The *loss function* is used to compare the reconstructed output with given input, evaluate their similarity and optimize the models parameters in order to minimize loss. In our experiments, we use binary cross entropy as a loss function. While cross entropy is used as a loss function in order to obtain suitable parameters of the autoencoder, mean square error is used as the main metric in order to judge the performance of the trained model and to choose the best performing one.

The autoencoder set up with aforementioned parameters is then trained using *batches* of training data of size 32. This means that instead of training and evaluating a network's performance with one instance per time, blocks of 32 instances are taken each time and the parameters of the network are updated with respect to all of them. Used batch size of 32 is a common parameter in training neural networks, as also suggested by Mishkin et al. (2016). More details on batch optimization as a common procedure in training deep neural networks can be found in Goodfellow et al. (2016).

An important aspect of training anomaly detection systems properly is the collection of network sub-flows for a period of time in order to train the appearance of normal flows. During the training phase, the system learns a simplified representation of a network sub-flow and tries to reconstruct the flow from this simplified representation. Then in operation, the error between reconstructed and original flow, called reconstruction error, is used as an indicator of a potential anomaly. In the training phase we observed that different protocols have different ranges of a reconstruction error. For this reason, we understood that

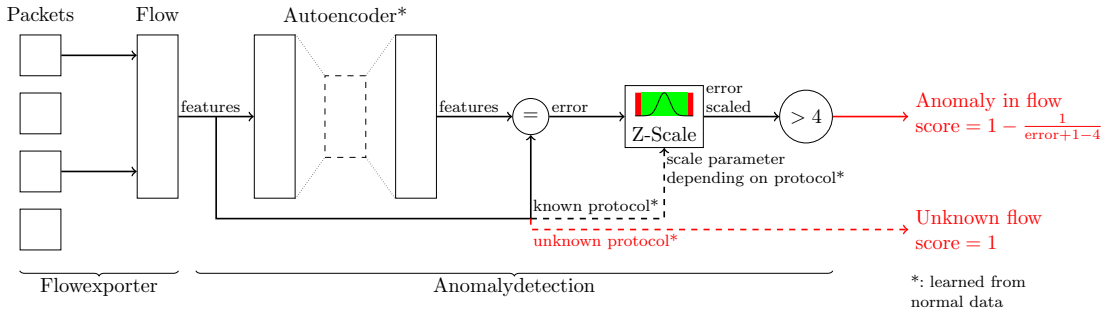

Figure 3: Overview of the network AD system.

establishing one threshold, as it is commonly done in anomaly detection systems, would not be suitable for our environment. Due to this reason, instead of one threshold, in the training phase we learn statistical properties and thresholds of each of the used protocols and then in the test time use this information to understand if an observed instance is far or close to the expected range. More precisely, in order to understand if an instance deviates from its expected range we use Z-normalization, also known as *normalization to zero mean and unit of energy* introduced by Goldin and Kanellakis (1995). This means that in the training phase for each of the protocols we calculate its mean and standard deviation. Then, in the test time, we Z-scale all instances of the known protocols (normalize them with previously recorded mean and standard deviation) and if they are more than four standard deviations distant, we declare them anomalous and calculate their anomaly score. We then perform scaling of the obtained anomaly score to the 0-1 range. This provides an explainable output, since we are informing the system user that detected instances with an anomaly score closer to 1 are more unusual (and hence suspicious) than those instances with anomaly scores closer to 0 from our detection system's point of view. On the other side, if we observe instances with unknown network protocols we give them the highest anomaly score of 1, since this is a clear sign of behavior not observed in the training phase.

It is important to point out that benign data is used for both training and testing of the system (no labels are needed and all the system's parameters were estimated directly from the benign data). Also, since our detection module focuses primarily on anomaly detection task, we opted for returning only the following two values: anomalous flow found (with certain anomaly score), and unknown protocol found. Normal flows are not output. An overview of the described, designed and deployed system for detection of network anomalies with an autoencoder at its core is depicted in Figure 3.

5.2. Anomaly detection on log data

For the purpose of detecting anomalies on log data, we apply the IDS ÆCID (Wurzenberger et al., 2018b), which stands for Automated Event Correlation for Incident Detection and monitors log data. ÆCID implements a self-learning approach that autonomously learns the normal system behavior and detects deviations from it. In this section we have a closer look on the concepts and components ÆCID uses. Figure 4 depicts ÆCID's process flow and visualizes the connection between the single components. ÆCID processes textual log data sequentially, i.e. line by line. It parses each log line, then applies several detection mechanisms and finally, if it detects an anomaly, it produces an output that describes the finding.

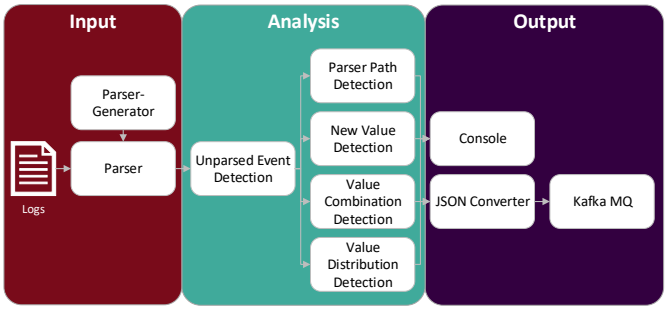

Figure 4: ÆCID pipeline.

5.2.1. Log line parsing

In a first step ÆCID parses log data. Log line parsers (He et al., 2016) are used to dissect log lines so that their content is accessible for further analysis. Furthermore, parsers allow to assign event types to log lines, which is necessary for certain types of log analysis. ÆCID offers two deployment options; one where data is analyzed on host side and one where the analysis is handled on a centralized network node. Especially the first one requires a highly efficient parser. For this purpose, common parsers that apply lists of regular expressions are insufficient, because they have a computational complexity of $O(n)$, where n is the number log event types. Thus, efficient log parsing that enables online anomaly detection would require large amounts of resources that are usually not available on many CPS components. Hence, ÆCID uses a novel parser approach that follows a tree-like structure. The complexity of parsing can be reduced to $O(log(n))$, which enables online anomaly detection on systems with low computational resources (Wurzenberger et al., 2019). The parser tree mainly consists of three building blocks: (i) static elements, (ii) variable elements and (iii) branch elements. Figure 5 provides an example of a parser for ntp logs. Furthermore, because of the tree-like structure each part of a parsed log line can be referenced using the path leading to the node, which supports further

analysis.

Since, services and components occurring in CPS usually do not provide log lines following any specific standard, there usually do not exist any predefined parser models. Furthermore, writing a detailed parser for all possible log lines is a cumbersome task. Thus, ÆCID provides a parser generator, ÆCID-PG (Wurzenberger et al., 2019) that allows to automatically generate parsers for log data of any syntax and grammar. In opposite to most other parser generators, ÆCID-PG does not rely on distance metrics. Instead, it follows a density-based approach (Vaarandi, 2003) that uses the frequency of tokens, i.e. delimited strings, occuring in log lines at specific locations.

5.2.2. Detecting anomalies

Since the ÆCID approach includes semi-supervised and unsupervised anomaly detection mechanisms, the process splits into a training and a detection phase. However, the training can be initiated and terminated at any time during runtime, which allows to extend the learned model of normal system behavior. Initially, ÆCID-PG is applied to learn the parser model. This can be done on pre-collected data that reflects the normal system behavior. Afterwards, ÆCID starts detecting anomalies. The parser itself builds the first layer of anomaly detection. Each internal system state relates to a specific log event type and each type of log event is represented by a path in the parser tree. In normal operation mode, usually only a small number of functionalities is used and the majority is not used. Hence, the parser implements a whitelist that allows all normal log events and reports log lines invoked by malicious and prohibited functionalities. Since, ÆCID-PG might have learned a parser model that inlcudes paths of anomalous log lines, a predefined hard-coded parser is used (for simple environments) or a parser generated on a similar system on which more different log lines occurred. Then, in a first phase ÆCID learns, which paths of the parser tree actually occur in the monitored system. Once, this phase is over, it starts reporting anomalies.

The first detector, *Unparsed Event Detection*, reports log lines that do not fit the parser model, i.e. there exists no path that represents this type of log event. Hence, this kind of anomaly is the most severe, since a functionality has been invoked or an action has been triggered that is not foreseen by the parser. The *Parser Path Detection* works similarly. It reports log lines that match paths of the parser, but have not been observed during the previously described second part of the training. These two detection mechanisms reveal anomalies related to usage of malicious or unwanted system functionalities or actions, which, for example, should only occur during maintenance, but not during normal system operation. In CPS such anomalies can be observed, for example, when an attacker initiates configuration changes.

The *New Value Detection* and the *Value Combination Detection* are also semi-supervised detection mechanisms and therefore require a training phase. Both build upon the parser and make use of the paths that lead to variable parts of the log lines. While the new value detection during the training learns all possible values that can occur in a certain locations in a log line, the new value detection learns combinations of variable values. After the training phase, ÆCID reports an anomaly every time it recognizes a value or a value combination that has not been monitored during the training. An anomaly related to a new value could, for example, be an IP address that has never before connected to a CPS device. A new value combination can be observed, e.g., when an administrator account carries out maintenance activities from an unusual machine. This would lead to an anomalous combination of username and IP address.

Finally, the *Value Distribution Detection*, is an unsupervised detection mechanism. Hence, it does not require a training phase. Similar to the new value detection and the value combination detection, this detector analysis the values occurring in log lines, where the parser allows variable input. The value distribution detection first decides for each variable node in the parser, if the values that occur in this location are static, ascending, descending, categorical/discrete, continuous or unique. Nodes that take unique values are not further analyzed. If the values are static, ascending or descending ÆCID raises an alarm, if the property of the values changes. An example for ascending values are IDs that increase continuously. If the value type is categorical/discrete or continuous, the value distribution detection learns the probability distribution and its parameters or the relative frequencies with which the values occur. If the detector recognizes any changes in the distribution, its parameters or the relative frequencies, it raises an alarm. Usernames and IP addresses are examples for categorical/discrete values. The backup user should, for example, occur less often than other users and might have a privilege level that could be exploited by an attacker. Continuous values occur especially in CPS and often relate to physical sensor values such as temperature, pressure or electric current. Furthermore, to reduce the number of false positives the value distribution detection offers an indicator function that ensures that an alarm is only raised when certain numbers of distributions fail within a specified time window.

5.2.3. Reporting anomalies

After the analysis, ÆCID offers several ways to report anomalies. The simplest one is to write the output to the console. However, more suitable for interfacing with other security solutions is to convert the output into JSON format and push it to a message queue such as Apache Kafka. This allows, for example, SIEMs to present ÆCID's findings to administrators and security analysts or to correlate it with information from other detection mechanisms or open source intelligence. ÆCID's output includes information on the host, where the anomaly has been detected, the time, when it has been detected, the raw log line, the detector that raised an alarm, the paths and the values related to the anomaly.

5.3. Cross-correlation and contextual enrichment in SIEMs

SIEM solutions are state of the art today, but they need to provide much more than simple query mechanisms and dashboards, if they are meant to be useful for continuous security monitoring. Powerful data analysis, in particular cross-correlation features as proposed in this paper, should be a key element of

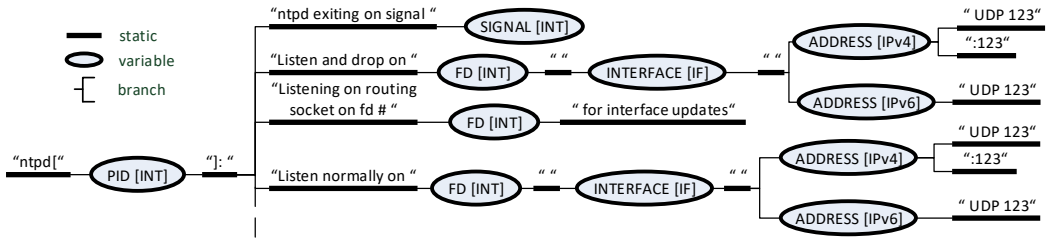

Figure 5: The tree describes the ÆCID parser model for ntpd (Network Time Protocol) service logs. Strings under quotes over bold lines are static elements. Oval entities allow variable values, bold lines mark static parts of the data and forks symbolize branches (Wurzenberger et al., 2018b).

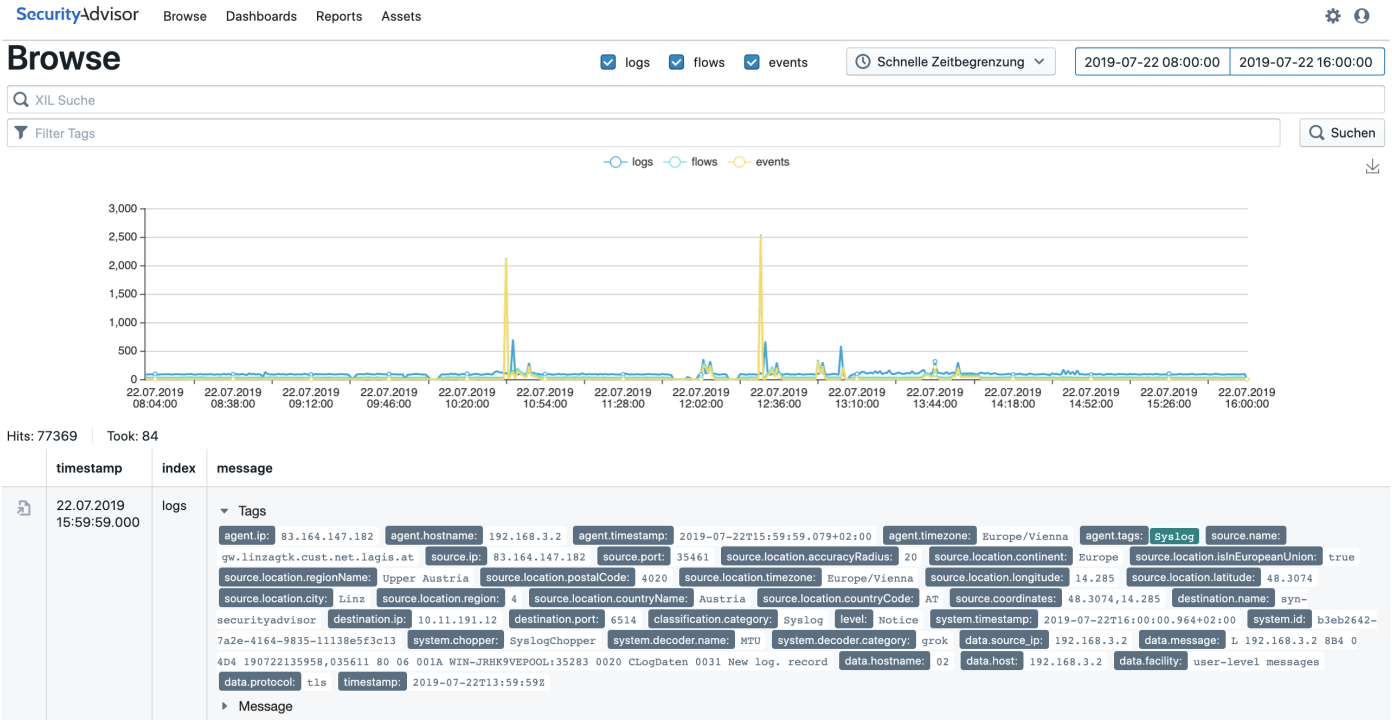

Figure 6: Screenshot of SecurityAdvisor’s (SA) full-text search.

CPS SIEMs to relieve the human in the loop. However, the benefit of a SIEM directly depends on the quality of the data that it is fed with. We propose that besides basic monitoring data and the anomaly detection results of network data AD and log data AD, as discussed in the sections before, further contextual data, including organizational and external sources, help tremendously to interpret AD results, gain insights into their root causes and rate their potential impact.

5.3.1. Technical integration of AD into SIEMs

The proposed synERGY approach digests network data, log data and higher-level event data (often generated from the SIEM solution itself using pre-defined rules for detecting known attack patterns). The integration of anomaly detection components into the synERGY architecture is quite simple thanks to the synERGY data broker (a Kafka message queue instance as outlined in the architecture (cf Sect. 4)). Any AD component can subscribe to numerous channels of interest (logs and network data of interesting segments of the observed CPS) and continuously fetches these raw operational data from the message queue. If anomalies are detected, they generate JSON-formatted messages following a predefined schema and push these into a predefined AD topic of the message queue. Figure 8 shows such a JSON-formatted sample alert that indicates a new co-occurrence of IP addresses in a particular log event. The SIEM solution takes the raw operational data together with the generated anomaly messages to further process both of them, i.e., aggregate and visualize raw data and cross-correlate generated alerts.

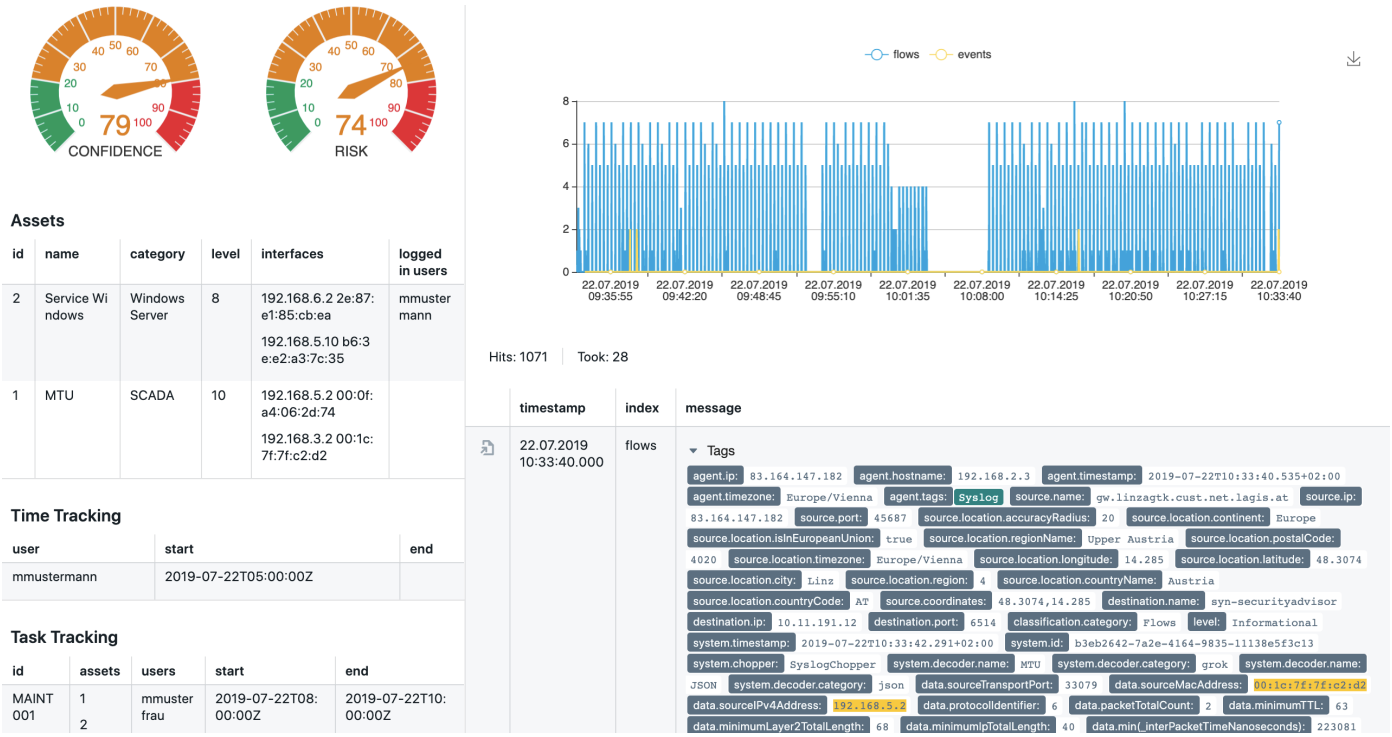

Figure 7: Screenshot of SecurityAdvisor's (SA) anomaly details.

```
{"Data":{
 "RawLogEvent":["Jul 22 12:28:05.839:
  %SEC-6-IPACCESSLOGNP: list 13 permitted 0
  192.168.5.1 -> 192.168.5.2, 4 packets"],
 "Time":[1563781058.819]},
"Anomaly":{
 "AffectedParserPaths":["acl/src", "acl/dst"],
 "Values":["192.168.5.1", "192.168.5.2"]},
"Detector":{
 "Type":"Combination Detector",
 "Description":"ACL IP Combo Detector"},
"Scores":[{
 "Type":"Severity",
 "Value":0.3}]}
```

Figure 8: Sample JSON alert for new IP address combination in log event.

5.3.2. *Cross-correlation of alerts*

The key to success here is that we correlate the results of the different AD mechanisms and increase confidence in findings if they match each other. In other words, if two or more distinct AD systems, which process entirely different kinds of data (or the same data but at least with different algorithms), come to the same conclusion, i.e., that there is an anomaly in a certain time frame concerning a certain system, it increases the likelihood of a relevant event and decreases the odds for a false positive alert. We use log data of (host) components to gain insights into their operational status and raw network data to observe communication patterns – and independently detect anomalies in these sources using different systems (namely tuna from Sect. 5.1 and ÆCID from Sect. 5.2).

AD systems usually produce alerts in a structured or semi-structured form, as outlined in Figure 8, which indicates a new co-occurrence of IP addresses in a particular log event. Based on this alert structure, it is possible to identify related anomaly events by comparing their attributes, e.g., find alerts in close temporal proximity that involve similar anomaly values. Selecting and weighting the attributes used for comparisons typically requires domain knowledge, but may be supported by mechanisms that rank attributes by their relevance, e.g., by computing their respective information gain entropy (Alhaj et al., 2016).

Applying more advanced cross-correlation techniques to detection results from different and in particular complementary AD systems enables improvement of quantitative security metrics regarding their accuracy and robustness. For example, the anomaly score visible in the sample alert in Fig. 8 could be increased if more alerts are found that concern the involved IP addresses. Thereby, it is common to combine several metrics derived from diverse alerts into a single or few values using numerical methods, for example, by computing a weighted average of all available risk scores (Alsubhi et al., 2008) or estimating detection confidences using statistical distributions (Roundy et al., 2017). Another approach is to consider alerts as time-series and aggregate their deviations to obtain an anomaly score that describes dynamic system behavior (Landauer et al., 2018).

We discuss the application of some of these approaches in our PoC in Sect. 6.3.3.

5.3.3. Interpretation of results using contextual elements

SIEMs are applied to interpret anomalies detected across system layers and across anomaly detection outputs in combination with organizational context (e.g. asset data, employee time recordings etc.) and open/external threat intelligence from vulnerability databases, mailing lists and online platforms.

Eventually, SIEMs are used to further correlate results on anomalies with information from risk management, configuration management, compliance management, etc. The particular challenge here is to apply the right data fusion methods suitable in CPS to limit the vast amount of information to a bearable quantity without losing important insights. Through applying the right mix of data fusion and contextualization approaches, detected anomalies can be interpreted more accurately, thus contributing to increasing situational awareness.

The *SecurityAdvisor (SA)*[6] used in our architecture, is a SIEM solution based on the well-known open-source ELK stack (Chhajed, 2015) that is used for collecting and correlating all relevant network-, log-, context- and anomaly data. This data is visually displayed inside a web-based, graphical user interface *(GUI)*, as shown in Fig. 6. For supporting the operator or analyst the 'Browse module' within the SA GUI offers various search, filter and visualization options, e.g. a full-text search which helps finding specific messages or events within the whole dataset. Besides the textual representation, the data is also displayed within a timeline, which makes it possible to detect outliers in the collected and analyzed data at first sight. Additionally, the operator can use a dashboard to create specific graphical representations (widgets) to gain further insights. The browse and dashboard modules are both useful for analyzing huge amounts of data and giving an overview of the whole dataset and infrastructure. For deeper analysis of a specific event, the SecurityAdvisor offers a 'Detail-View', as shown in Figure 7. This view displays a confidence and risk score, data related to the affected assets, and contextual data, e.g. data collected from the organisation's time-tracking and task-tracking databases. For a deeper understanding of the anomaly, all the data (network traffic, log) which triggered the anomaly, is correlated by the backend, and displayed to the operator.

Several mechanisms for alert correlation and aggregation have been proposed as outlined in Sect. 2. Our applied aggregation technique relies on the computation of a weighted average of the single results reported by tuna (cf. Sect. 5.1) and ÆCID (cf. Sect. 5.2) respectively (whereas the weights can be tuned based on operator preferences and experience (Bouzar-Benlabiod et al., 2010)). The resulting value is further scaled by a risk value that the system calculates by taking context information (mainly asset information) into account as follows:

The displayed confidence value corresponds to the anomaly score given by the anomaly detection systems, whereas the shown risk score is calculated by the SA, using contextual elements within the following schema:

- *Confidence*

  The confidence level of the anomaly detection system is taken into account with max. 50%, which means 100% confidence = 50% risk score.

- *Asset Risk*

  The asset risk is filling up the other 50% and is itself based on the following elements:

  - *Asset Level*

    Each asset is assigned a criticality level within the asset management database. This level is taken into account with a maximum of 33.3%.

  - *Asset User*

    The SA tries to identify the currently logged-in user on the affected asset, which is only possible if the system is using an operating system which supports users, e.g. Windows or Linux. If a logged-in user can be identified, it is possible to include further context data into the risk score calculation, such as time tracking. Then the user is looked up in the connected time-tracking system and determined whether the user is present or not (e.g. on vacation). If the user is not present, the anomaly is classified as higher risk, and therefore the asset risk is raised by 33.3%. If the user is not found, the time-tracking system can not be checked, so the asset risk is automatically raised by 33.3%.

  - *Task Tracking*

    For each affected asset the task-tracking database is being checked for an open maintenance order and schedule. The check is based on the asset, the user (if found earlier) and the timestamp of the detected anomaly. If no entry in the task-tracking system is found, the asset risk is raised by 33,3%.

An analyst or operator must assume, that the anomaly is more critical for higher risk scores, because no relevant context data was found to justify the anomaly and subsequently lowering the risk score.

6. Implementation and Proof-of-Concept

In this section we describe the technical realization and results of our proof-of-concept evaluation.

6.1. Implementation

The deployment of the synERGY proof-of-concept is separated into three areas: (i) SCADA data, (ii) context data, and (iii) SIEM & anomaly detection. Figure 9 shows an overview of the contents and interfaces between these three areas. In the following, we will describe each area individually.

[6] http://www.securityadvisor.eu [last accessed on 01/10/2020]

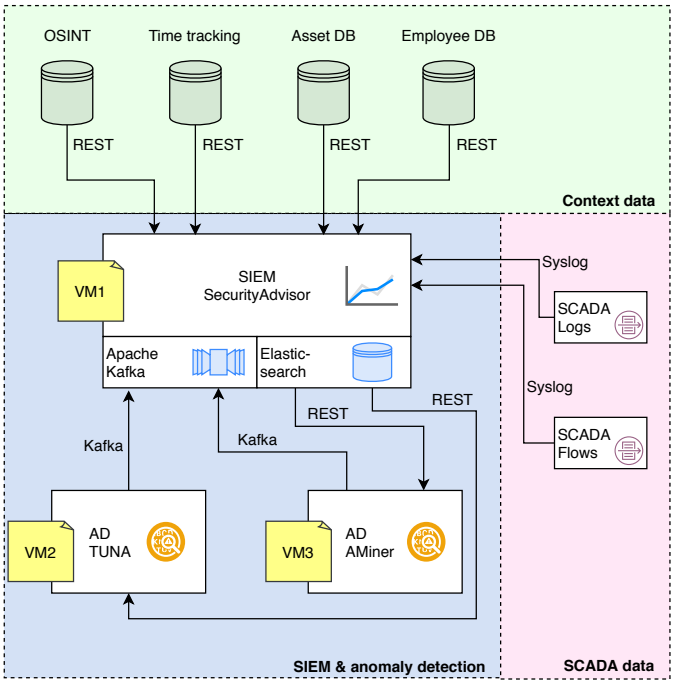

Figure 9: Deployment of the synERGY architecture.

6.1.1. SCADA data

This area involves the collection of log and flow data from all relevant components present in the synERGY infrastructure (cf. Sect. 3.2). In particular, we used flows from a network tap and logs from a SCADA server that monitors and controls components such as actuators, an Identity Services Engine (ISE), a firewall, a switch, and a primary substation's MTU that collects and aggregates log messages from all connected secondary substations' RTUs. Note that most of these components do not support installation of agents for log data collection. Therefore, we use the Syslog protocol to transmit the data to our SIEM.

6.1.2. Context data

Context data is technical and corporate information gathered by an enterprise or organization. For our use case, we consider the following types of context data: (i) Open Source INTelligence (OSINT) that includes publicly available information on cyber threats, (ii) time tracking records that keep track of the location of employees at particular times, e.g., their presence in the office or at a remote location, (iii) Assets that include all components and devices present in the network, and (iv) Employee information such as names, permissions, and roles. The data are stored on separate databases and accessed by the SIEM over a Representational State Transfer (REST) interface.

6.1.3. SIEM & anomaly detection

Our SIEM system SecurityAdvisor is the central component of our deployed infrastructure. The SIEM queries context data over a REST interface, receives flow and log data through Syslog, and stores the obtained data in two separate indices in an ElasticSearch[7] database. Note that whenever the syntax of the received data is known, it is possible to extract and store particular information in separate fields, e.g., the Syslog protocol requires that the generation timestamp as well as the hostname of the component that generated the message is transmitted for each log line. This allows querying the ElasticSearch database for particular attributes, e.g., retrieve all events that occurred in a certain time range or were generated by a specific device. Unstructured information such as the actual log message that does not follow any known structure are stored in raw format in the ElasticSearch database.

While some SIEMs include advanced analysis systems in addition to visualizations, we decided to separate our anomaly detection systems in favor of a modular approach that allows us to develop our algorithms independent from each other. We therefore separated the deployment of the SecurityAdvisor, the flow-based anomaly detection system TUNA (that implements the mechanisms described in Sect. 5.1) as well as the log-based anomaly detection system ÆCID/AMiner (see Sect. 5.2) on different Virtual Machines (VM). The anomaly detection systems query the stored events using the ElasticSearch REST API, disclose anomalous system behavior, and put anomaly events on a Kafka message queue, which are then consumed and displayed by the SecurityAdvisor GUI.

6.2. Proof-of-Concept

We evaluated the anomaly detection capabilities of our approach in the deployment scenario outlined in the previous section. For this, we collected network traffic and log data for several days while the proof-of-concept system was idle, i.e., no actions were executed by humans and only automatic processes generated log data. We used this data to train our anomaly detection systems before deploying them for detection. We then carried out several experiments based on the attack types outlined in Sect. 3. In the following, we describe one particular test case in detail and then discuss the results of the detection.

The test scenario assumes that an attacker was able to obtain a technician's notebook and credentials that can be used to connect to the MTU over the network (cf. Sect. 3). The attacker logs into the system and proceeds to change the configuration of the substation. Finally, the attacker restarts the MTU so that the changes become active, and logs out of the system. This attack scenario involves several steps that are expected to manifest themselves in log and flow data. In particular, the following list details the points in time where the malicious actions were performed:

- 12:28: Enable network adapter (windowsUp, windowsActive)
- 12:31: Log in (serviceLogin, serviceActive)
- 12:33: Download, modify, and upload changed configuration (configDown, configUp)

[7] https://www.elastic.co [last accessed on 12/23/2019]

- 12:35: MTU restart (mtuReboot)
- 12:37: Logout and disconnect network interface (windowsDown)

With this attack scenario, we expected to address several attack types listed in Table 1 that are suitable for either log-based or network-based anomaly detection. To begin with, the notebook used by the attacker is a 'newly added device' identified by its IP address. Moreover, the attacker carries out 'configuration changes' on the substation followed by an unscheduled 'device boot up'. From a network point of view, these actions cause anomalous 'traffic statistics' that are subject to detection. Furthermore, they generate new events at suspicious points in time that should be recognizable through log-based anomaly detection. Finally, the actions of the attacker do not comply with normal user behavior and should thus be detectable by 'comparison of traffic profiles'.

6.3. Results

We executed the attack on the deployed system as outlined in the previous section. In order to adequately evaluate the capabilities of our anomaly detection tools, we analyzed their detection performance separately. In the following, we discuss properties of successfully detected attack manifestations and explain why some malicious actions remained undetected.

6.3.1. Network Traffic Anomaly Detection

Since the analyzed network data does not contain packet payloads (but just flow meta data; see Sect. 5.1) and the anomalies generated in the tests were in part only assignable on the basis of the payload data, an analysis of the original network packet data is performed based on expert knowledge and anomalies, which were resulting from the performed test procedure. Since the packet-to-subflow mapping is unambiguous, the packet data could be associated to the recorded flow data, the packet anomalies were transferred to flow anomalies, and then compared against the results of the anomaly detection.

Figure 10 shows how our network anomaly detector performed in the designed use case. The top plot shows the timing of total and anomalous flow occurrences over time, the center plot visualizes detected anomalies per time unit, and the bottom plot visualizes undetected anomalies per time unit. Thereby, the anomalies are divided into individual categories. In addition to the data mentioned above, the respective recorded times are displayed with vertical lines. The time differences between recorded times and occurrence of the anomalies results from the minute-minute time recording of the events and from delays required by equipment and software, e.g., duration of booting. The categories serviceActive and windowsActive are used, since both the connection service program with MTU, as well as Windows itself (for example, DNS requests), consistently cause network traffic while they are active. Some events in the graph are shown with very bright areas (e.g., mtuReboot, windowsUp), because these events generally trigger very few anomalous flows. Login events are included in the network data, but could not be identified from the network packets due to unknown protocols, which is why this anomaly is part of windowsActive. Overall, Figure 10 shows that all but windowsDown events were detected by network anomaly detection. This is because removing the computer from the network does not generate network traffic.

Our detection system performs to a high satisfactory level on most of the tested events. However, there are still some cases where not all anomalous events could be detected. Following is the list of them together with explanations on why such anomalies could not be detected in higher number.

- windowsDown: These are not answered address resolution protocol (ARP) requests sent from the firewall. They could not be detected due to normal ARP requests.
- windowsActive: ARP Requests that are normal and could only be detected based on MAC/internet protocol (IP) address (that we do not analyze).
- serviceLogin: Messages exchanged via SCADA protocol that look normal and could only be detected via deep packet inspection.
- mtuReboot: Same situation as with serviceLogin.

6.3.2. Log Data Anomaly Detection

In contrast to flow data, it is possible to access log messages in raw format. However, labeling log lines automatically was not feasible due to the complex and diverse content of the lines. In addition, log lines that correspond to adversarial behavior often differ only slightly or not at all from normal behavior. Therefore, the logs and disclosed anomalies were reviewed manually using expert knowledge.

Figure 11 shows the results of our log-based anomaly detection system. The top plot displays the total number of generated log lines as well as the total number of disclosed anomalies aggregated in intervals of 5 seconds. As visible in the plot, most anomalies occur in close temporal proximity to some of the attack steps, and it is thus likely that they are related to the malicious behavior. A more detailed view is provided in the bottom plot, that separates detected anomalies according to their origin. The coloring displays the ratio between the amount of anomalies and total amount of log messages, where darker colors mean that more anomalies were disclosed in relation to the total amount of logs, and vice versa.

A majority of anomalies are unparsed logs, i.e., log messages with unknown syntax (cf. Sect. 5.2). These anomalies are caused by new events that did not occur during the training phase. In our test scenario, adversarial actions frequently caused the detection of unparsed logs generated by the MTU, such as logs that indicate connection establishment and authentication at 12:31 or logs generated during reboot starting at 12:35.

Another important type of anomalies concern new values or new combinations of values in correctly parsed log messages. In our test scenario, the IP address of the attacker was detected several times and by different log sources, i.e., the anomalies

Figure 10: Network traffic anomaly detection performance in the test case scenario.

in the switch and firewall logs between 12:28 – 12:31. An exemplary alert representing a new occurrence of IP addresses is shown in Fig. 8. Finally, new parser paths were detected between 12:35 – 12:36 in the SCADA logs. They refer to log lines related to time synchronization that did not occur during training, but could be parsed since the syntax of their messages is known.

Overall, windowsUp, attacker login, as well as MTU reboot were successfully detected. There were no false positives among the disclosed anomalies. We manually searched the logs for manifestations of the configuration upload and service logout, but could not find any corresponding messages. In order to enable the generation and in turn detection of these log messages it is necessary to set a higher logging level. In Sect. 7, we will outline the issue of an insufficient logging level in more detail.

6.3.3. Alert Correlation

In the previous sections, we presented the detection results of a log-based and a flow-based AD system obtained from a specific use-case. As visible in Fig. 10 and Fig. 11 respectively, several injected attack steps were successfully detected by both systems, making it reasonable to consider their results in a combined form to gain a more extensive view of the attack and its effects on the monitored system. As outlined in Sect. 5.3, there exist several possibilities to correlate alerts on different layers. In the following, we will discuss the application of a selection of correlation techniques that are suitable for our data.

We first consider the groups of anomalies detected at 12:28 (windowsUp) and 12:31 (login). As mentioned before, the involved log-based anomalies are triggered by the occurrences of new IP addresses. The flow-based AD system on the other hand did not consider IP addresses for detection, since they were dropped during learning; instead, statistical deviations within the traffic meta-data caused the anomalies. Nevertheless, the IP addresses explicitly identified as anomalous by the log-based system (cf. Fig. 8 that specifies "Values" in alert) are suitable to be matched with IP addresses in flow attributes. Successful matches of such attributes in different alerts occurring in short time windows increase the overall risk score displayed in the SIEM, because their combined occurrence is a better indicator for actual anomalies than the individual alerts alone. For example, new IP addresses could regularly occur in dynamic networks and trigger false positives, and statistical chance could cause that normal network traffic associated with the same IP is incorrectly classified as anomalous; however, their combined occurrence in close temporal proximity decreases the probability for such misclassifications.

Another factor that contributes to the resulting overall risk score related to these anomalies is provided by available context information. In our case, the IP used by the attacker is associated with a specific notebook of a technician. We configured our SIEM to automatically look up all attributes disclosed as anomalous in the asset and user databases. As part of our scenario, we made sure that these queries yield the information that the technician is not supposed to be working during the time of attack. This makes the observed activities even more suspicious and thus further increases the risk score according to the calculation schema stated in Sect. 5.3.

Since the MTU configuration changes and user logout are not detected by both AD systems and thus risk score and confidence are directly derived from the reported alerts without further calculations, we focus on the reboot of the MTU starting at 12:35 as another exemplary case for alert correlation. As out-

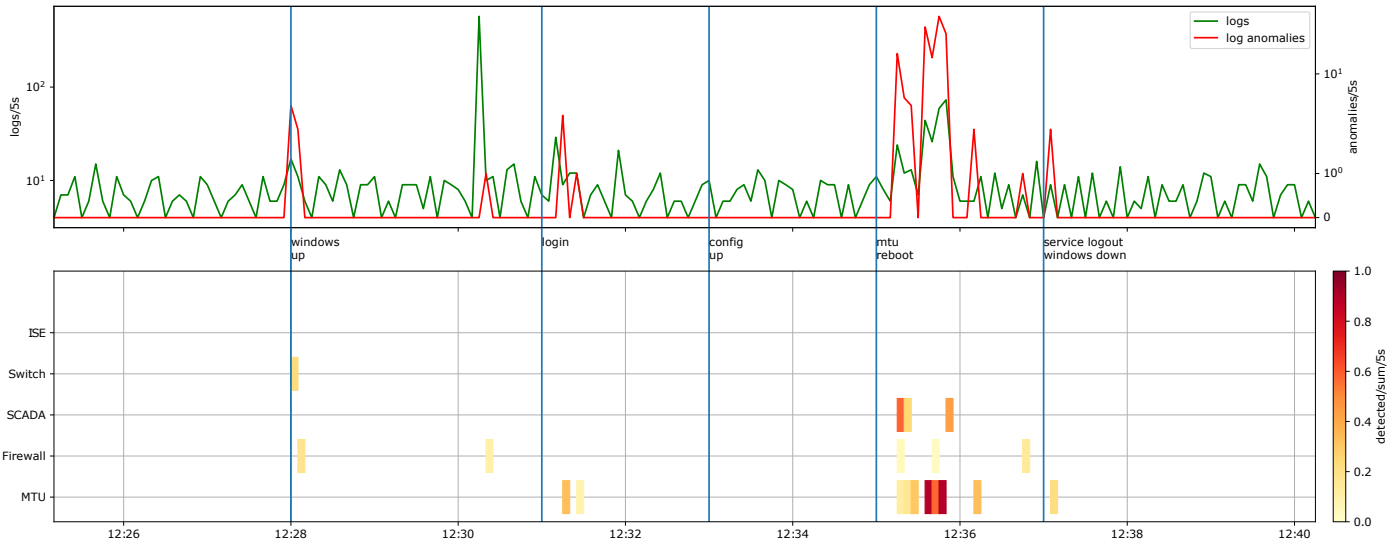

Figure 11: Log data anomaly detection performance in the test case scenario.

lined in the previous section, the anomalous log lines generated during this attack step could not be parsed, meaning that no attributes such as IP addresses could be extracted for matching. For this reason, we rely only on time-series analysis to find related alerts. As visible in the plots, the increase of unparsed log lines in the MTU log file and anomalous network traffic associated with the MTU occur almost simultaneously, meaning that the respective time-series have positive correlation and the origin of the anomalies detected by both systems are likely to have the same origin. We use this to compute an overall risk score that is the average of all risk scores reported by the detectors and again increase the resulting confidence of the alert.

7. Critical discussion and economic analysis

At this point, let us discuss results obtained from synERGY. We provide our lessons learned that comprise helpful insights for anyone planning to carry out a project in the field of anomaly detection for system security. In addition, we outline the economic analysis of the application scenarios from synERGY.

7.1. Lessons learned from the pilot application

Building a realistic testing environment for our use-case scenarios that also supports the integration and application of our anomaly detection and correlation tools was non-trivial. During development, we faced a number of issues that forced us to re-design parts of our infrastructure and its configuration. In the following, we share some of the pitfalls that may concern the application of log-based anomaly detection in general, highlight limitations of our solution, and outline suggestions for improvement.

7.1.1. Pitfalls of real application

A majority of challenges stem from the setup of the logging infrastructure and the configuration of the components that contribute to the collected logs. Due to the fact that the outcomes of all detection and analysis tools rely on the quality of these logs, it is of utmost importance to define a setup that is both realistic and usable for the application scenario. This mainly concerns the verbosity of the logs, i.e., the amount of information that is logged, which is configured by the logging level. By default, many components are configured to only produce error and warning logs for the sake of reducing the amount of storage space for the logs. However, these alert logs are already a kind of anomalies themselves; unless they occur with such regularity that they actually document permanently repeating system behavior (which should be investigated and fixed by admins), they are an inappropriate source for any kind of anomaly detection. The reason for this is that unsupervised anomaly detection relies on the assumption that anomalies are few in comparison to the whole data set (Chandola et al. (2009)). Therefore, informational or debugging logging levels are required to produce logs that document the normal system behavior and enable the generation of behavior models usable for the detection of suspicious log events within that data. Since the desire to produce these fine-grained logs is usually met with resistance from operators due to limitations of storage capacity, we suggest to apply anomaly detection tools that process each log line only once at their generation, but do not require the whole data to be stored permanently.

Even adequately configured logging levels do not guarantee that the generated logs are suitable to be used for detection. For example, while all other components produce logs in the range of seconds, the switch (Process Network in Fig. 1) aggregates all occurring events in time windows and prints a summary only every few minutes. This causes a delay in the detection of anomalous events from this log source, impedes counting events in arbitrary time windows for event-frequency-based anomaly detection, and complicates the derivation of correlation rules spanning over multiple components, e.g., event A observed at component X implies that event B has to occur at component Y within a certain amount of time. If technically

possible, logs should therefore be produced at the finest available resolution and as close to real-time as possible.

Another problem is the generation of realistic normal behavior, that is essential for the evaluation of any anomaly detection tool. Monitoring the system in a completely idle state is suitable to collect information on its background noise, e.g., logs that are produced by automatically scheduled tasks such as cron jobs. However, models trained only on this data are likely to suffer from high false positive rates, because almost every manually executed action on the system is recognized as an anomaly. We therefore suggest to first observe the behavior of a productive system, derive and define the main aspects of what is considered normal behavior, and finally implement a script that simulates user interaction and can be executed during training to generate more representative normal system behavior than background noise alone.

We also experienced difficulties with the interpretation of logs as well as artificially raised anomalies. Domain knowledge on the setup and purpose of the components in the system environment emerged as essential information for understanding the processes running on the machines as well as their manifestations in the logs. However, this knowledge is often not available in the organization running the machines, or not accessible to the analysts. This lack of ground truth makes manual validation of derived models after training as well as evaluation of the detection capabilities problematic. In particular, a differentiation of logs that correspond to normal behavior during attack phases and logs that are known to correspond to actual malicious behavior would be needed. One solution is to shift from log-based evaluation, where every log event is considered, towards a less detailed time-window-based evaluation, where only time windows that contain anomalies are considered.

Finally, logs are high-volume data produced in fast rates, thus requiring extensive resources for computation and transmission. Lack of high-performance computing machines will inevitably lead to delays or even breakdowns. We recommend to deploy machines that can handle the high loads.

7.1.2. Detection capabilities

Our experimental results suggest that log-based and flow-based anomaly detection tools are suitable to detect the applied attacks. Log-based detection is in particular effective for the disclosure of new events, i.e., events that have not been observed in the training phase and are thus not part of the parser model. On the other hand, detection based on concrete parameter values is a promising technique for disclosing more stealthy attacks with minimal manifestations, but is also subject to higher false positive rates, because parameter values are usually more likely to show random fluctuations. This situation improves when the detection scope is manually limited to certain parameters, combinations of parameters, or events, when configuring the detectors.

Furthermore, our experiments showed that anomalies reported by two different AD systems are suitable for correlation. In particular, we were aiming to provide system administrators two values that represent an aggregated risk level of the whole system and the trust in this prediction, as visible in the top left of Fig. 7. Thereby, our applied aggregation technique (cf. Sect. 5.3) relies on the computation of a weighted average of the reported anomalies as well as several other input values from diverse sources, including contextual databases providing asset risk estimations. However, since contextual risk scores are relatively static, it is clear that the expressiveness of the aggregated risk score and confidence strongly depend on the validity of the anomalies and their severity classifications reported by the deployed AD systems. In the following, we discuss some issues that we faced with this matter and suggest possible solutions.

Among the drawbacks of our log-based detection tool is that a anomaly score can only be assigned to certain types of anomalies. The reason for this is that it is difficult to extract semantic information from the anomalies, e.g., it is not trivial to decide whether a new IP occurring as a parameter value depicts a more or less severe violation of the learned model than a sequence of log events occurring in incorrect order, or any other form of anomaly. One possible solution is to manually assign a weight to specific detectors, for example, we consider unknown events as the most severe anomalies due to the fact that they violate the structure of the known log format, which means that it is not possible to relate them to any known event, making manual review always necessary. Another possibility is to incorporate external information to compute the score, for example, organizational context data (cf. Sect. 3).

Similar to logs, we assign the maximum anomaly score to flows of unknown protocols. Other than that, the structured format and availability of semantic information within flow data makes computing an anomaly score simpler than for log data, because the divergence to expected flows can be numerically expressed through statistical methods.

When it comes to our proposed network analysis approach, it is important to understand how frequently the detection system has to be fully updated to remain representative (and thus provide high detection performance). The exact answer to such a requirement is strongly dependent on the application-specific scenarios where the systems are used. In general, anytime it is observed that network traffic used for training is not fully representative of the expected benign behavior of the monitored system, the detection system would profit from a repeated training phase in which instances of newly observed benign behavior are taken into account.

Another set of challenges for successful usage of the designed network traffic analyzer is related to the selection of suitable parameters and their fine-tuning. In our detection scenario we experimented with different values for detection parameters, but most of these are dataset and use-case specific, and accordingly the system would need to be retrained once placed in a new detection environment.

7.1.3. Maintenance effort

Systems are subject to change over time. On the one hand, components are removed and added to the network, on the other hand, software updates may affect the logging behavior. In any way, self-learning anomaly detection tools are only partially able to adapt to such changes. We pursue the following approach: When an anomaly is detected, e.g., a new IP address

occurs as a parameter value, an anomaly is raised and the analyst is given the opportunity to declare it as benign (a false positive) or malicious. In case that the event is benign, the IP address will be white-listed, i.e., it is added to the model of normal behavior and will not be detected in the future. This reduces the effort to handle false positives to a minimum. However, analysts are also required to regularly check whether entries in the white-lists become outdated, and adapt them if necessary. Otherwise, attacks could be purposefully designed to conform to these outdated white-lists, thus circumventing detection.

7.2. Economic analysis of cross-correlation

This section describes exemplary economic analysis results for different application areas of attack detection. Examples of such possible application areas are listed below:

- Protection of individual systems (e.g. photovoltaics or data concentrators)
- Protection of individual grid sections (e.g. hydro-power plants; avoidance of power outages)
- Protection of entire networks (e.g. avoidance of power outages in the city of Linz or Upper Austria)

To evaluate application areas of attack detection economically, the achievable benefit is put in relation to the expected costs (investment and operating costs of the installed sensors). In this respect, the following cost factors are of central importance: (i) Capital and operational expenditures of sensors, and (ii) operational expenditures (OPEX) of false alarms.

Capital and operational expenditures of sensors: To estimate the possible applications for the detection of attacks, the costs of the components used must be compared with the achievable advantages (e.g. avoidance of a power failure). The total investment costs of the technologies used (e.g. Test Access Ports or Switched Port Analyser) are made up of the direct and indirect investment costs. Direct investment costs refer to an operating resource itself (e.g. installed sensor), indirect investment costs cover the other one-off costs associated with the respective operating resources (e.g. installation, replacement of components, etc.).

These total investment costs (as the sum of the direct and indirect investment costs) are finally converted into annual payments by multiplication with the annuity factor. To these annual capital expenditures (CAPEX) Hirth (2017) in a further step the annual OPEX of the used resources are added Dugmore and Lacy (2006).

OPEX of false alarms: As with any statistical test, false positive and false negative alerts are ultimately unavoidable. The economic cost-benefit ratio achieved by synERGY to some extent depends on how the system is embedded in organizational processes, and how much disruption, and thereby implied cost, a false alert may cause. Estimations of these costs are based on error rates related to sensor detection accuracies (false-positive rates of sensors), as well as hard- and software reliability, and also the ratio of human-error in decision making. We conducted a cost analysis based on the following assumed (simplified) alert treatment process that starts with a sensor sending

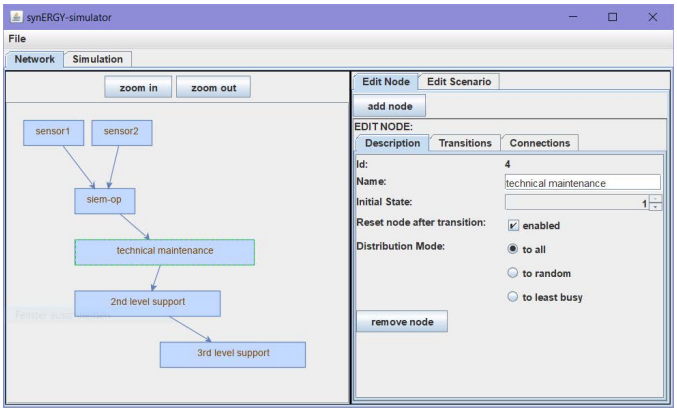

Figure 12: Alert treatment cost simulator (example screenshot).

an anomaly notification. Upon arrival of an alarm, the operator of the SIEM system first decides on the potential presence of a real or a false alarm (triggering of the sensor despite regular behavior of the system, indicated by further sensors or implied by the overall system behavior). If the alarm is suspected to be real, the report is sent to a person in the technical administration who will examine the problem on site (costs for travel and on-site investigation of the cause of the alarm). Otherwise, there will be no further action or involvement of personnel. The technician checks the cause of the fault, which refutes the suspicion of an incident (a "false-positive" that the SIEM operator could not recognize), or there is a fault in the sensor, which the technician rectifies. These costs add to the operative costs in the evaluation, whereby their variation results in bandwidths of the cost-benefit ratio.

While this presumed treatment is rather generic (and as such admits even a simple analytic computation of the costs based on a probability analysis using the underlying workflow that is essentially like a decision tree), the synERGY project features a customizable simulation component (see Figure 12) to model any more general and arbitrarily complex workflow of alert treatment, including signals from multiple sensors, multiple SIEM operators, queue-based task assignments to groups (following a first-come first served or a random service regime), multi-level support structures and several others. The inner model used to simulate the treatment is a system of coupled automata and an event-based simulation, adapted from Schauer et al. (2019).

7.2.1. Exemplary economic analysis results

The previously mentioned bandwidths are calculated via application-specific break-even points Wöhe et al. (2016) considering low and high cost (CAPEX and OPEX). A break-even point in this context represents the intersection of the cost (e.g. low cost if sensors are realized by Switched Port Analysers) and benefit function for a specific attack detection application (e.g. protection of individual power plants). It furthermore indicates, how often an attack must be fended off in the respective application area to cover the costs through the achievable advantages. For the exemplary areas of attack detection discussed above, the results shown in Figure 13 (which is based on results

of the synERGY project) apply. According to the performed calculations, the left edges of the bars represent a low-cost implementation (CAPEX and OPEX) as well as the right edges a high-cost implementation (CAPEX and OPEX) of the attack detection system. It can be seen that the protection of individual systems would require successful attack defense within the time bandwidth of weeks (protection of Photovoltaic plants) to months (protection of data concentrators). This bandwidth increases to several years or a decade if high economic losses caused by power failures (e.g. in Linz or Upper Austria) can be avoided.

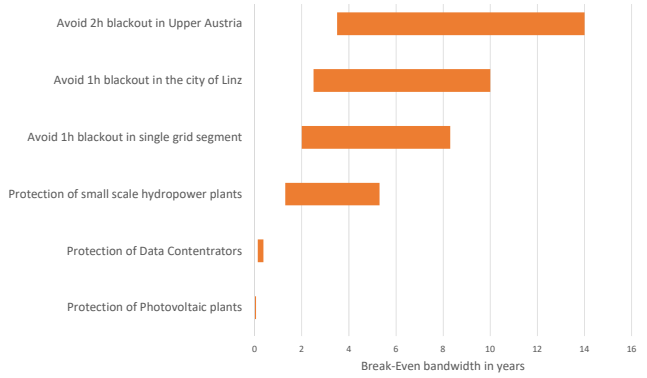

Figure 13: Cost-benefit analysis results.

7.2.2. Derived findings

The actual number of attacks fended off during the period under consideration has a significant influence on the cost-effectiveness of attack detection. There is therefore a risk of high running costs if the attacks are not carried out with sufficient frequency. The height of the CAPEX for the sensors used has a very strong effect on the economic efficiency of the system. The higher the CAPEX, the less any OPEX variations have an effect on the break-even points. It does not seem realistic to build a system to protect individual components, as many attacks per year would have to be detected and averted to cover the resulting costs. If, however, significant benefits are achieved (e.g. by avoiding large-scale power failures and the associated economic costs), an economic operation is reachable for a corresponding frequency of attacks.

8. Conclusion and future work

In this paper, we reported on the result of a three year's research project centered on intrusion detection in cyber-physical systems. We described the differences to common enterprise IT and introduced an architecture making use of numerous anomaly detection systems capable of processing the various types of data (logs, network data, event streams etc.) in a CPS that can enable effective attack detection and interpretation (through the use of contextual elements). Our main contribution is not about the core anomaly detection methods, but showing the various ways on how the integration of different components (even from different sources) could work to enable cross-correlation and a demonstration of the added value. In order to realize the synERGY approach, we implemented a proof-of-concept system following the designed architecture, and challenged it in a realistic, yet illustrative, way to show its potential with respect to intrusion detection in a utility provider's infrastructure. We discussed the numerous pitfalls when it comes to the implementation and deployment of such an AD system and highlighted the importance of cross-correlating AD results from distinct detection systems, as well as the correlation with organizational context to enable the proper interpretation of security events and reinforce the significance of reports.

Even after the three year's project there remain numerous open research challenges. Smaller near-term goals relate to how such a system can be set into operations more efficiently, i.e., create parsers, rules and set parameters with minimum human involvement. This is important to lower the entry barrier for interested stakeholders of the synERGY system. Currently, experts are required to validate and tune the parsers for complex log data, train the different machine learning models and come up with initial rule sets for the detection engines. The medium-term goal is to make the operational phase of the synERGY system economically feasible by involving even more organizational knowledge to interpret anomalies more accurately, which is the basis for justified decision making. Our long-term goals are centered on improving synERGY's ability to adapt to changes in the monitored environment, which means that the detection and interpretation must be changed too (including, models, learned parameters, weights etc.) – preferably in a (semi-)automatic manner.

Considering future advanced use cases of cyber-physical systems with potentially large attack surfaces combined with high criticality for the general public, especially in the smart grid area (Skopik, 2014), the application of a system like synERGY gains even more importance.

Acknowledgments

This work was partly funded by the information and communication technology (ICT) of the future program in course of the project synERGY (855457).

References

Aburomman, A. A., Reaz, M. B. I., 2017. A survey of intrusion detection systems based on ensemble and hybrid classifiers. Computers & Security 65, 135–152.

Alhaj, T. A., Siraj, M. M., Zainal, A., Elshoush, H. T., Elhaj, F., 2016. Feature selection using information gain for improved structural-based alert correlation. PloS one 11 (11).

Alsubhi, K., Al-Shaer, E., Boutaba, R., 2008. Alert prioritization in intrusion detection systems. In: NOMS 2008-2008 IEEE Network Operations and Management Symposium. IEEE, pp. 33–40.

Arcaini, P., Riccobene, E., Scandurra, P., 2015. Modeling and analyzing mape-k feedback loops for self-adaptation. In: 2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems. IEEE, pp. 13–23.

Bolla, R., Comi, P. M., Repetto, M., 2018. A distributed cyber-security framework for heterogeneous environments. In: ITASEC.

Bouzar-Benlabiod, L., Benferhat, S., Boubana-Tebibel, T., 2010. Integrating security operator knowledge and preferences to the alert correlation process. In: 2010 International Conference on Machine and Web Intelligence. IEEE, pp. 416–420.

Bray, T., December 2017. The javascript object notation (json) data interchange format. RFC 8259, RFC Editor.

Chandola, V., Banerjee, A., Kumar, V., 2009. Anomaly detection: A survey. ACM computing surveys (CSUR) 41 (3), 15.

Chhajed, S., 2015. Learning ELK Stack. Packt Publishing Ltd.

Chuvakin, A., Schmidt, K., Phillips, C., 2012. Logging and log management: The authoritative guide to understanding the concepts surrounding logging and log management. Newnes.

Claise, B., Trammell, B., Aitken, P., September 2013. Specification of the ip flow information export (ipfix) protocol for the exchange of flow information. RFC 7011, RFC Editor.

Debar, H., Wespi, A., 2001. Aggregation and correlation of intrusion-detection alerts. In: International Workshop on Recent Advances in Intrusion Detection. Springer, pp. 85–103.

Dugmore, J., Lacy, S., 2006. A Managers' Guide to Service Management. BSI Standards.

Elshoush, H. T., Osman, I. M., 2011. Alert correlation in collaborative intelligent intrusion detection systems—a survey. Applied Soft Computing 11 (7), 4349–4365.

European Council, 2016. Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46/ec (general data protection regulation). URL https://eur-lex.europa.eu/eli/reg/2016/679/oj

Fernandes, G., Rodrigues, J. J. P. C., Carvalho, L. F., Al-Muhtadi, J. F., Proença, M. L., Mar 2019. A comprehensive survey on network anomaly detection. Telecommunication Systems 70 (3), 447–489. URL https://doi.org/10.1007/s11235-018-0475-8

Friedberg, I., Skopik, F., Settanni, G., Fiedler, R., 2015. Combating advanced persistent threats: From network event correlation to incident detection. Computers & Security 48, 35–57.

Garcia-Teodoro, P., Diaz-Verdejo, J., Maciá-Fernández, G., Vázquez, E., 2009. Anomaly-based network intrusion detection: Techniques, systems and challenges. computers & security 28 (1), 18–28.

Gerhards, R., 2009. The syslog protocol. RFC5424.

Goh, J., Adepu, S., Tan, M., Lee, Z. S., 2017. Anomaly detection in cyber physical systems using recurrent neural networks. In: 2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE). IEEE, pp. 140–145.

Goldin, D. Q., Kanellakis, P. C., 1995. On similarity queries for time-series data: constraint specification and implementation. In: International Conference on Principles and Practice of Constraint Programming. Springer, pp. 137–153.

Goldstein, M., Uchida, S., 2016. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. PloS one 11 (4), e0152173.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, http://www.deeplearningbook.org.

Gu, G., Cárdenas, A. A., Lee, W., 2008. Principled reasoning and practical applications of alert fusion in intrusion detection systems. In: Proceedings of the 2008 ACM symposium on Information, computer and communications security. pp. 136–147.

Gupta, S., 2012. Logging and monitoring to detect network intrusions and compliance violations in the environment. URL https://www.sans.org/reading-room/whitepapers/detection/logging-monitoring-detect-network-intrusions-compliance-violations-environment-33985

Han, S., Xie, M., Chen, H.-H., Ling, Y., 2014. Intrusion detection in cyber-physical systems: Techniques and challenges. IEEE systems journal 8 (4), 1052–1062.

Harada, Y., Yamagata, Y., Mizuno, O., Choi, E.-H., 2017. Log-based anomaly detection of cps using a statistical method. In: 2017 8th International Workshop on Empirical Software Engineering in Practice (IWESEP). IEEE, pp. 1–6.

He, P., Zhu, J., He, S., Li, J., Lyu, M. R., 2016. An evaluation study on log parsing and its use in log mining. In: 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). IEEE, pp. 654–661.

Hemsley, K. E., Fisher, E., et al., 2018. History of industrial control system cyber incidents. Tech. rep., Idaho National Lab.(INL), Idaho Falls, ID (United States).

Hernández, L., Baladron, C., Aguiar, J. M., Carro, B., Sanchez-Esguevillas, A., Lloret, J., Chinarro, D., Gomez-Sanz, J. J., Cook, D., 2013. A multi-agent system architecture for smart grid management and forecasting of energy demand in virtual power plants. IEEE Communications Magazine 51 (1), 106–113.

Hirth, H., 2017. Grundzüge der Finanzierung und Investition. Walter de Gruyter GmbH & Co KG.

Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., Pras, A., Fourthquarter 2014. Flow monitoring explained: From packet capture to data analysis with netflow and ipfix. IEEE Communications Surveys Tutorials 16 (4), 2037–2064.

Hu, S., Tang, C., Liu, F., Wang, X., 2016. A distributed and efficient system architecture for smart home. International Journal of Sensor Networks 20 (2), 119–130.

Humayed, A., Lin, J., Li, F., Luo, B., 2017. Cyber-physical systems security—a survey. IEEE Internet of Things Journal 4 (6), 1802–1831.

Hyvärinen, A., Karhunen, J., Oja, E., 2001. Independent component analysis, adaptive and learning systems for signal processing, communications, and control. John Wiley & Sons, Inc 1, 11–14.

Inoue, J., Yamagata, Y., Chen, Y., Poskitt, C. M., Sun, J., 2017. Anomaly detection for a water treatment system using unsupervised machine learning. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW). IEEE, pp. 1058–1065.

Javaid, A., Niyaz, Q., Sun, W., Alam, M., 2016. A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS). ICST (Institute for Computer Sciences, Social-Informatics and . . . , pp. 21–26.

Ji, C., Li, Y., Qiu, W., Awada, U., Li, K., 2012. Big data processing in cloud computing environments. In: 2012 12th international symposium on pervasive systems, algorithms and networks. IEEE, pp. 17–23.

Jolliffe, I., 2011. Principal component analysis. Springer.

Junejo, K. N., Goh, J., 2016. Behaviour-based attack detection and classification in cyber physical systems using machine learning. In: Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security. pp. 34–43.

Kammerstetter, M., Langer, L., Skopik, F., Kupzog, F., Kastner, W., 2014. Practical risk assessment using a cumulative smart grid model. In: SMARTGREENS 2014 - Proceedings of the 3rd International Conference on Smart Grids and Green IT Systems. ACM, pp. 31–42.

Kent, K., Souppaya, M., 2006. Guide to computer security log management. URL https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-92.pdf

Krishnamurthy, S., Sarkar, S., Tewari, A., 2014. Scalable anomaly detection and isolation in cyber-physical systems using bayesian networks. In: ASME 2014 Dynamic Systems and Control Conference. American Society of Mechanical Engineers Digital Collection.

Kubiak, P., Rass, S., 2018. An Overview of Data-Driven Techniques for IT-Service-Management. IEEE Access 6, 63664–63688. URL https://doi.org/10.1109\%2Faccess.2018.2875975

Kwon, Y., Kim, H. K., Lim, Y. H., Lim, J. I., 2015. A behavior-based intrusion detection technique for smart grid infrastructure. In: 2015 IEEE Eindhoven PowerTech. IEEE, pp. 1–6.

Landauer, M., Wurzenberger, M., Skopik, F., Settanni, G., Filzmoser, P., 2018. Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. computers & security 79, 94–116.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. nature 521 (7553), 436–444.

Lee, J., Bagheri, B., Kao, H.-A., 2015. A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufacturing letters 3, 18–23.

Löf, A., Nelson, R., Sep. 2014. Annotating network trace data for anomaly detection research. In: 39th Annual IEEE Conference on Local Computer Networks Workshops. pp. 679–684.

Li, D., Jayaweera, S. K., 2014. Distributed smart-home decision-making in a hierarchical interactive smart grid architecture. IEEE Transactions on Parallel and Distributed Systems 26 (1), 75–84.

Liao, H.-J., Lin, C.-H. R., Lin, Y.-C., Tung, K.-Y., 2013. Intrusion detection system: A comprehensive review. Journal of Network and Computer Applications 36 (1), 16–24.

Liu, C., Ghosal, S., Jiang, Z., Sarkar, S., 2016. An unsupervised spatiotemporal graphical modeling approach to anomaly detection in distributed cps. In: 2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS). IEEE, pp. 1–10.

Lu, T., Lin, J., Zhao, L., Li, Y., Peng, Y., 2015. A security architecture in cyber-physical systems: security theories, analysis, simulation and application fields. International Journal of Security and Its Applications 9 (7), 1–16.

Ma, Z., Hudic, A., Shaaban, A., Plosz, S., 2017. Security viewpoint in a reference architecture model for cyber-physical production systems. In: 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). IEEE, pp. 153–159.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y., 2018. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Computing 17 (3), 12–22.

Micheel, J., Donnelly, S., Graham, I., 2001. Precision timestamping of network packets. In: Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement. pp. 273–277.

Mishkin, D., Sergievskiy, N., Matas, J., 2016. Systematic evaluation of cnn advances on the imagenet. arXiv preprint arXiv:1606.02228.

Mitchell, R., Chen, I.-R., 2014. A survey of intrusion detection techniques for cyber-physical systems. ACM Computing Surveys (CSUR) 46 (4), 1–29.

Morin, B., Mé, L., Debar, H., Ducassé, M., 2009. A logic-based model to support alert correlation in intrusion detection. Information Fusion 10 (4), 285–299.

Moustafa, N., Slay, J., 2015. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 military communications and information systems conference (MilCIS). IEEE, pp. 1–6.

Overman, T. M., Sackman, R. W., 2010. High assurance smart grid: Smart grid control systems communications architecture. In: 2010 First IEEE International Conference on Smart Grid Communications. IEEE, pp. 19–24.

Rass, S., 2017. On Game-Theoretic Risk Management (Part Three) - Modeling and Applications. ArXiv e-prints.
URL `http://arxiv.org/pdf/1711.00708`

Rass, S., Zhu, Q., 2016. GADAPT: A Sequential Game-Theoretic Framework for Designing Defense-in-Depth Strategies Against Advanced Persistent Threats. In: Zhu, Q., Alpcan, T., Panaousis, E., Tambe, M., Casey, W. (Eds.), Decision and Game Theory for Security. Vol. 9996 of Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 314–326.

Roundy, K. A., Tamersoy, A., Spertus, M., Hart, M., Kats, D., Dell'Amico, M., Scott, R., 2017. Smoke detector: cross-product intrusion detection with weak indicators. In: Proceedings of the 33rd Annual Computer Security Applications Conference. pp. 200–211.

Sabahi, F., Movaghar, A., 2008. Intrusion detection: A survey. In: Systems and Networks Communications, 2008. ICSNC'08. 3rd International Conference on. IEEE, pp. 23–26.

Sadooghi, I., Palur, S., Anthony, A., Kapur, I., Belagodu, K., Purandare, P., Ramamurty, K., Wang, K., Raicu, I., 2014. Achieving efficient distributed scheduling with message queues in the cloud for many-task computing and high-performance computing. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, pp. 404–413.

Sakurada, M., Yairi, T., 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In: Proceedings of the MLSDA 2014 2nd Workshop on Machine Learning for Sensory Data Analysis. ACM, p. 4.

Scarfone, K., Mell, P., 2007. Guide to intrusion detection and prevention systems (idps). NIST special publication 800 (2007), 94.

Schauer, S., Grafenauer, T., König, S., Warum, M., Rass, S., 2019. Estimating cascading effects in cyber-physical critical infrastructures. In: International Conference on Critical Information Infrastructures Security. Springer, pp. 43–56.

Settanni, G., Skopik, F., Karaj, A., Wurzenberger, M., Fiedler, R., 2018. Protecting cyber physical production systems using anomaly detection to enable self-adaptation. In: 2018 IEEE Industrial Cyber-Physical Systems (ICPS). IEEE, pp. 173–180.

Settanni, G., Skopik, F., Shovgenya, Y., Fiedler, R., Carolan, M., Conroy, D., Boettinger, K., Gall, M., Brost, G., Ponchel, C., et al., 2017. A collaborative cyber incident management system for european interconnected critical infrastructures. Journal of Information Security and Applications 34, 166–182.

Shaaban, A. M., Kristen, E., Schmittner, C., 2018. Application of iec 62443 for iot components. In: International Conference on Computer Safety, Reliability, and Security. Springer, pp. 214–223.

Shakeri, M., Shayestegan, M., Abunima, H., Reza, S. S., Akhtaruzzaman, M., Alamoud, A., Sopian, K., Amin, N., 2017. An intelligent system architecture in home energy management systems (hems) for efficient demand response in smart grid. Energy and Buildings 138, 154–164.

Skopik, F., 2014. The social smart grid: Dealing with constrained energy resources through social coordination. Journal of Systems and Software 89, 3–18.

Skopik, F., Ma, Z., Bleier, T., Grüneis, H., 2012. A survey on threats and vulnerabilities in smart metering infrastructures. International Journal of Smart Grid and Clean Energy 1 (1), 22–28.

Skopik, F., Settanni, G., Fiedler, R., 2016. A problem shared is a problem halved: A survey on the dimensions of collective cyber defense through security information sharing. Computers & Security 60, 154–176.

Tanenbaum, A. S., Van Steen, M., 2007. Distributed systems: principles and paradigms. Prentice-Hall.

Vaarandi, R., 2003. A data clustering algorithm for mining patterns from event logs. In: Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003)(IEEE Cat. No. 03EX764). IEEE, pp. 119–126.

Valdes, A., Macwan, R., Backes, M., 2016. Anomaly detection in electrical substation circuits via unsupervised machine learning. In: 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI). IEEE, pp. 500–505.

Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R. A., 2004. Comprehensive approach to intrusion detection alert correlation. IEEE Transactions on dependable and secure computing 1 (3), 146–169.

Velan, P., Čermák, M., Čeleda, P., Drašar, M., 2015. A survey of methods for encrypted traffic classification and analysis. International Journal of Network Management 25 (5), 355–374.

Wachter, J., Grafenauer, T., Rass, S., 2017. Visual Risk Specification and Aggregation. In: SECURWARE 2017: The Eleventh International Conference on Emerging Security Information, Systems and Technologies. IARIA, pp. 93–98.

Wickramasinghe, C. S., Marino, D. L., Amarasinghe, K., Manic, M., 2018. Generalization of deep learning for cyber-physical system security: A survey. In: IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society. IEEE, pp. 745–751.

Wöhe, G., Döring, U., Brösel, G., 2016. Einführung in die allgemeine Betriebswirtschaftslehre. Vahlen, Franz.

Wurzenberger, M., Landauer, M., Skopik, F., Kastner, W., 2019. Aecid-pg: A tree-based log parser generator to enable log analysis. In: 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM). IEEE, pp. 7–12.

Wurzenberger, M., Skopik, F., Settanni, G., 2018a. Big data for cybersecurity. In: Sakr, S., Zomaya, A. (Eds.), Encyclopedia of Big Data Technologies. Springer International Publishing, Cham, pp. 1–9.
URL `https://doi.org/10.1007/978-3-319-63962-8\_163-1`

Wurzenberger, M., Skopik, F., Settanni, G., Fiedler, R., 2018b. Aecid: A self-learning anomaly detection approach based on light-weight log parser models. In: ICISSP. pp. 386–397.

Yuan, D., Park, S., Zhou, Y., 2012. Characterizing logging practices in open-source software. In: Proceedings of the 34th International Conference on Software Engineering. IEEE Press, pp. 102–112.

Yuan, Y., Jia, K., 2015. A distributed anomaly detection method of operation energy consumption using smart meter data. In: 2015 International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP). IEEE, pp. 310–313.

Zhou, C., Paffenroth, R. C., 2017. Anomaly detection with robust deep autoencoders. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp. 665–674.

ACL access control list

AD anomaly detection

ARP address resolution protocol

CAPEX capital expenditures

CPS cyber-physical systems

DB database

DHCP dynamic host configuration protocol

DoS denial-of-service

HIDS host based intrusion detection system

ICT information and communication technology

IDS intrusion detection system

IP internet protocol

IPFIX IP Flow Information Export

ISE Identity Services Engine

IT information technology

JSON Java Script Object Notation

LAN local area network

MAC media access control

MTU Master Terminal Unit

NIC network interface card

NIDS network based intrusion detection system

OPEX operational expenditures

OSINT Open Source INTelligence

OT operational technology

REST Representational State Transfer

RTU Remote Terminal Unit

SCADA supervisory control and data acquisition

SIEM security information and event management

SMS short message service

SNMP simple network management protocol

SPAN switched port analyzer

SPA stateful protocol analysis

SSH secure shell

TACACS terminal access controller access control system

TAP terminal access point

TCP transmission control protocol

TLS Transport Layer Security

VLAN virtual local area network

WAN wide area network