Behavior-Based Anomaly Detection in Log Data of Physical Access Control Systems

Florian Skopik, *Senior Member, IEEE*, Markus Wurzenberger, Georg Höld, Max Landauer, and Walter Kuhn

Abstract—Behavior-based anomaly detection (AD) approaches for enterprise-IT security are not easily applicable to other domains, such as embedded devices and IoT nodes in cyber-physical systems. AD approaches are usually highly optimized for specific purposes, tightly bound to domain-specific technologies and rely on a specific syntax of investigated data. Data from cyber-physical systems is however highly diverse, often poorly documented and not easily ingested for automated analysis. AECID provides an anomaly detection approach, that monitors unstructured textual event data (i.e., log data), and implements self-learning for autonomous operation. A parser generator establishes a model of normal system behavior on top of observed events, which then can be leveraged to detect anomalies as deviations from that baseline. The unsupervised anomaly detection approaches of AECID apply machine learning techniques to perform sequence analysis, correlation analysis and statistical tests of events represented in log data. This paper discusses AECID's applicability in a building security system use case. A proof of concept demonstrates the effective detection of anomalies in log data of a building access control system stemming from card misuse, including stolen access cards and cloned cards.

Index Terms—Access control, anomaly detection, behavior modeling, intrusion detection, log data, machine learning, security

1 Introduction

Enterprise IT, embedded systems, smart manufacturing, energy grids, industrial IoT, fintech, and other domains, operate interconnected systems, which follow predefined processes and are employed according to specific usage policies. The events generated by the systems governed by these processes are usually recorded as log data [1] for maintenance, accountability, or auditing purposes. Such records contain valuable information that can be leveraged to detect any inconsistency or deviation in the process, and indicate anomalies potentially caused by attacks, misconfiguration or component failure. The investigation of such events can be performed adopting different anomaly detection (AD) techniques [2].

AD approaches have been largely investigated by IT security researchers and have been proven effective for very specific problems; nevertheless, these methods have not been fully leveraged to achieve security in other domains, yet. The reasons for this are manifold, but a major hurdle is that log data is highly diverse and their structure mostly poorly documented [3]. Instead of a few major system types like in enterprise IT, in operational technologies, IoT and cyber-physical systems (CPS) hundreds of vendors exist, which makes it even harder to maintain an understanding of system log data structures. Additionally, the life span of CPS components is much longer than in enterprise IT, which usually leads to a mixture of devices of different types and generations in large CPS. These are all factors that hinder the wide adoption of standards for log data in this area and leads to highly diverse and often insufficiently documented log output [4].

We argue that the advantages provided by AD in enterprise IT security should efficiently be transferred to other domains, including embedded systems and IoT security as applied in building security systems (BSS). Specifically, methods from user entity behavior analytics (UEBA) [5], used to determine fraudulent behavior on enterprise endpoints, are a boon to CPS by determining behavior profiles of entities within a CPS – and pinpoint any kinds of behavior anomalies on top of that. In this paper, we take a closer look into the application of AD in physical access control systems [6]. Here, key cards allow users to enter areas of a building for which they have been authorized. By determining deviations from individually learned usage patterns of cards using log data from building security equipment, we demonstrate that we are able to discover different kinds of fraud, such as stolen cards and cloned cards, and other kinds of weaknesses, including excessive access rights.

The main contribution of this paper is the demonstration of a smart anomaly detection approach in a non-enterprise IT environment. The detection approach was designed for self-learning a baseline in an unsupervised fashion and applies subsequent online anomaly detection of deviating behavior. Specifically, our goal is to detect strong changes of the card holders' behavior regarding their booking activities

- *Florian Skopik, Markus Wurzenberger, Georg Höld, and Max Landauer are with the Center for Digital Safety and Security, AIT Austrian Institute of Technology, 1210 Vienna, Austria. E-mail: {florian.skopik, markus.wurzenberger, georg.hoeld, max.landauer}@ait.ac.at.*
- *Walter Kuhn is with PKE Holding AG, Technologie und Systeme, 1100 Vienna, Austria. E-mail: walter.kuhn@chello.at.*

Manuscript received 12 November 2021; revised 2 June 2022; accepted 4 August 2022. Date of publication 8 August 2022; date of current version 11 July 2023. This work was supported by the Austrian FFG research program ICT of the Future with the project DECEPT 873980.
(Corresponding author: Florian Skopik.)
Digital Object Identifier no. 10.1109/TDSC.2022.3197265

as a consequence of fraudulent actions. We further discuss the feasibility of four different detectors for physical access control, where we investigate appropriate means to make them applicable in a real life setting with noisy data due to erratic users in changing environments.

We apply AECID [7], a cutting edge machine learning solution that flexibly adapts to the observed infrastructure, reliably builds system behavior models, and predicts and detects behavior changes caused by intrusion attempts, unintended configuration changes, and malfunction. We show how AECID can be applied to increase the protection provided by an existing facility security control center.

The remainder of the paper is organized as follows. Section 2 provides important background on physical security systems and use cases for AD in this context. Then, Section 3 outlines the AD methodology of AECID. Following these design principles, we highlight the implementation for physical access control in Section 4. Specifically, we show the structure of log data and the configuration of suitable detectors. In Section 5, we demonstrate the application on real data from a medium-sized operational environment. Then, Section 6 reports on important lessons learned and critically discusses limitations of our work. Eventually, Section 7 contains related work and Section 8 concludes the paper.

2 Building Security Systems

A multitude of data sources exist in building security systems (BSS), suitable for detecting behavioral anomalies of its users. In this paper we mainly focus on the usage of access cards in physical access control, however, also data from IoT cameras or other means of sensors are potentially suitable to achieve similar results. Depending on the use case, different types of anomalies, such as outliers or change in trends need to be detected.

2.1 Anomaly Detection (AD) in BSS

The primary goal of Intrusion Detection Systems (IDS) is to timely detect invaders and attackers, to react quickly and reduce the caused damage [8]. There are many parallels in how IDS are used in the areas of 'classic' enterprise IT security and building security systems. While, for example, firewalls and antivirus scans are applied to detect and prevent attacks in computer networks, physical access control systems (such as electronic locks) and surveillance cameras are used for the same purposes in physical facilities.

Smart anomaly detection approaches are required to reliably discover deviations from a desired system's behavior because of an unusual utilization through an illegitimate user [2]. Notice that this is the same regardless if a high-end server landscape is being exploited or a low-end embedded system of a BSS. For instance, if adversaries manage to steal user credentials, or access cards in case of building security, and are using these legitimate credentials to illegitimately access systems they will eventually utilize them differently from legitimate users. In IT environments, they run scans, search shared directories and try to extend their presence to surrounding systems. In physical building, they may do the same, i.e., systematically try out where they can get access or moving through a building on unusual routes. This will generate a series of events identifiable by anomaly-based detection approaches. For that purpose, a baseline describes the 'normal and desired system behavior' and everything that differs from this description is classified as hostile. This way, AD approaches can find new attack vectors that signature-based approaches cannot.

2.2 Use Cases of AD in BSS

We describe four use cases of AD for physical access control, to demonstrate the benefits of AD. Detectors for these use cases are further implemented and evaluated in the remainder of this paper. Notice, certain misuse cases are not covered by the introduced detectors, e.g., piggybacking, for which other means, such as information from surveillance cameras, would need to be processed. Additionally, the less the attacker knows about the legitimate card holder and/or the company policies the more likely they will cause anomalies. In contrast. The better an attacker can mimic the legitimate card holder's behavior, the harder it will be for the system to discover misuse.

2.2.1 UC#1 Systematic Trial and Error

A casual attacker finds a physical access card and enters a building. Within the building s/he systematically tries out to which rooms s/he can get access to with this card within short time intervals. This usage pattern is unusual for the legitimate owner of the card. A high number of booking events in small time slots occur, potentially outside previous usage periods, and at doors usually not used in this sequence. Numerous types of anomalies may occur (unusual frequencies, breaking sequences etc.), leading to detection of this misuse case.

2.2.2 UC#2 Cloned Cards

Similar to UC#1 an attacker gets hold of a legitimate access card, however, this time s/he clones the card leading to two identical cards causing booking events in the access system. The same detection mechanisms as before apply. Moreover, this time the AD system has an additional sophisticated means to detect misuse, since two identical cards are now used, potentially at close points in time. The AD system could observe booking events across all cards at the different doors of a building and built an understanding of how close doors are located and how much time is needed to progress from one door to another one. If the legitimate owner uses his card at one part of the building, while the attacker causes bookings in a another one, temporal conditions will be violated.

2.2.3 UC#3 Excessive Access Rights

In this case, a card has been granted access rights to a room, but was never used to open the corresponding door. This happens quite frequently if access rights are not managed individually but via group memberships. Although the legitimate card owner would be allowed to go through the door the usage on that particular lock would be highly suspicious. An anomaly in this case is thus a booking event at a door by a card holder who has never (or rarely) crossed this door.

2.2.4 UC#4 Remote Unlock Operations

Several BSS offer the operator the ability to temporarily unlock doors for all persons remotely. This is a feature to

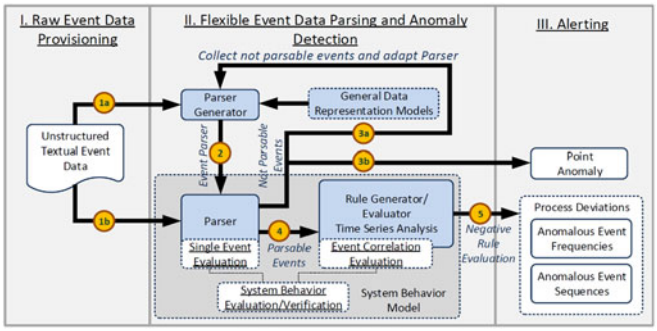


Fig. 1. Approach to anomaly detection in log data.

ease delivery of goods. In many cases specific doors are unlocked through predefined schedules at fixed time slots or days of a week. For exceptional cases, operators can issue unlock operations manually too. If an attacker bribes or blackmails an operator, unusual unlock operations, e.g., at doors that were never unlocked in the past or unlock operations at uncommon times, would be detected by AD.

3 Anomaly Detection With AECID

This section provides some background information on applicable log data analysis techniques and subsequent anomaly detection in CPS.

3.1 Anomaly Detection in Logs Beyond Enterprise IT

While AD approaches for IT security have been investigated by researchers for years, this field has not been deeply explored beyond enterprise IT, especially for small cost-effective devices with limited computing power, such as low-end embedded systems and IoT nodes. Signature and block-listing based approaches are only capable of detecting malicious single events, instead, AD allows also to monitor complex processes and to detect attacks that manifest in malicious event frequencies and sequences [9]. However, due to high false positive rates, challenges during parsing unstructured textual log lines and fast changes in the syntax of the log data, carrying out AD is not trivial. The rapidly changing cyber threat landscape and the high data diversity, caused by a high variety of computer systems/IT devices and frequently changing networks, require flexible and self-adaptive anomaly detection approaches. Self-learning solutions build a baseline of the normal system behavior during the training phase, and then use it as ground truth to detect anomalies that expose attacks in the detection phase [10].

Typical machine (deep) learning approaches such as artificial neural networks (ANN), Bayesian networks, decision trees, hidden Markov models (HMM) and support vector machines (SVM) often require large amounts of resources that are, as in case of embedded systems and IoT, often not available. Moreover, the lack of appropriate and well-balanced trainings data is an issue. Furthermore, they mostly work for numerical data only and not for rather complex interdependent text data, such as computer log data. Reasons for this are high dimensionality of the data (compared to simple numeric series), a lack of intuitive distance metrics, and encoding issues [11], [12]. Hence, AECID applies rather traditional methods from statistics and machine learning. Clustering [13], and methods from probability [14] and graph theory [15] are leveraged to identify significant properties of log data and to build efficient parser models which allow to reduce computational complexity of log parsing. Furthermore, procedures adopt stochastic methods, such as statistical tests, stochastic processes and association rules to model normal system behavior and to identify rules which describe unique event sequences and correlations in log data.

3.2 Overview of the AECID Approach

We apply the anomaly detection (AD) approach AECID[1], that was designed to work on textual log data. Fig. 1 illustrates the operational mode of the AECID methodology. The approach analyzes unstructured textual event data, such as syslog messages from computer systems or protocol data from manually recorded events (e.g., access logs).

In the training phase (1a) a parser generator analyzes textual log data and automatically builds event parsers, i.e., identifies implicit structures in apparently unstructured records using clustering, to decompose and understand textual representations of events with no manual intervention [16]. After the training phase (2), the parser obtains the event parsers from the parser generator. The parser analyzes then the unstructured textual event data entities separately (1b), i.e., it performs a single event evaluation. Thus, depending on the configuration, the parser either forwards not parseable events to the parser generator – which collects them and adapts the event parsers (3a) -, or it triggers a point anomaly (3b). The rule generator/evaluator and time series analysis module obtain parseable events from the parser, and it defines and evaluates statistical rules. For example, it evaluates the distribution with which events occur, it defines observed good event sequences, and it carries out a time series analysis. Thus, the module detects deviations of complex processes from the normal system behavior as anomalous event frequencies and sequences (5). Eventually, the parser along with the rule evaluator module, define a model of the normal behavior of the observed environment (e.g., a physical access control system to a building), evaluate the model and continuously verify it.

4 System Implementation

We discuss the implementation of an unsupervised anomaly detection system for a BSS using the AECID approach outlined before. In particular, we investigate how log data is being parsed and analyzed with a set of specifically configured detectors.

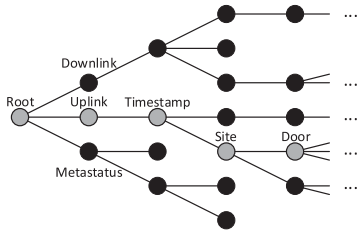
4.1 Monitoring and Logging

Log data is the lowest common denominator of data that any piece of software can produce to inform about its operational state. Thus, log data is a key information source for many different applications, such as anomaly detection. We collect log data from a building security system, specifically from the physical access module of a AVASYS.access system,[2] to steer the development of the analytical model that captures the baseline of normal system behavior. The

1. AMiner: https://github.com/ait-aecid/logdata-anomaly-miner [7]
2. PKE AVASYS.access with ADC-M and ADC-S controllers installed in PKE's headquarter in Vienna

Downlink;2021-01-01 07:00:00,000;AT VIE: Entrance garage (1F);ACS door-controller 11/door 1 – Operating status;permanent unlock;-(CEN);. . .
Uplink;2021-01-01 07:12:36,000;AT VIE: Entrance PKE (1F);ACS door-controller 10/door 1/reader 1 – status;06:20:36 valid card card541 person558;. . .
Uplink;2021-01-01 07:20:36,000;AT VIE: Main entrance 3F;ACS door-controller 24/door 1/reader 1 – status;06:21:36 invalid card card175 person197;-(SSD.1);. . .
Downlink;2021-01-01 07:21:19,000;AT VIE: Main entrance 2F;ACS door-controller 8/door 2 – Operating status;door unlock;operator043;. . .
Uplink;2021-01-01 07:25:10,000;AT VIE: Service entrance (BSMT);ACS door-controller 9/door 6/reader 1 – status;06:25:10 valid card card717 person742;. . .

(a) Example for log lines of door transitions and remote unlocks of doors. The down/uplink specifies the direction in which the information was transmitted.



(b) Example for a parser.

Path	Path value
/Uplink/Timestamp/Site/Door	ACS door-controller 9/door 6/reader 1
/Uplink/Timestamp/.../AccessText/Card	card717

(c) Example for parser paths and corresponding path values.

Fig. 2. Example for the parsing process: raw log lines are dissected according to a parser tree to make single log tokens accessible via parser paths for further analysis.

investigated log data includes log lines of a variety of event types, such as updates and connection status. For the analysis, only two event types are of particular interest. The first type relates to door opening events and indicates if the access to a door $d \in D$ was granted or rejected for a card $c \in C$. The second type consists of remote unlock operations, where an operator $o \in O$ remotely granted access to a door d or the door d was unlocked according to a schedule. Fig. 2a provides example lines for both event types.

4.2 Logdata Parsing

Log data occurs in form of unstructured or semi-structured text lines that describe a certain system or network event. Thus, log parsing is an important task prior to log analysis. A log parser defines the syntax, i.e., unique structure, of the data produced by a monitored system or service. It describes log lines as series of nodes and maps elements of log lines to these nodes, e.g., by assigning strings separated by white spaces in text messages to nodes. The nodes restrict the benign format or type of values, e.g., only allowing a static string, numerical values, timestamps of a certain structure, etc. This way, the log parser gives semantic meaning to nodes. AECID uses a specific type of parser that has a tree-like structure (see Fig. 2b). The tree-like structure reduces the complexity of log line parsing and thus improves performance [16].

Fig. 2 depicts the parsing process of an example log line. The parser dissects the framed log line from left to right by traversing the parser tree and successively mapping substrings of the log line to its nodes. The traversal starts at the root node of the parser tree and accounts for the mentioned node restrictions. This way the parser process ensures that the whole log line is syntactically correct and mapped to a known parser path. Each value of a parser node eventually describes a part of the original log line. In the further analysis phase, each token is referenced by a unique path. Two paths and the values of the log line are listed in Fig. 2c. The path consists of a sequence of node identifiers which specify the traversal route to the node in the parser (similar to a directory structure in a file system) and the value is the corresponding substring in the log line. This way, we can efficiently access the value of a door d and a card c (among others) in a logged event without the need of common linear tokenization. For the details of this approach please refer to [16].

4.3 System Behavior Model Building

In this paper, we focus on two basic event types. The first one captures the usage of a card $c \in C$ (usually through its legitimate owner) to open a door $d \in D$; the second one logs an operator's $o \in O$ remote unlock operations (see UC#4) of a specific door d. The recorded log data $L = [l_1, l_2, \ldots]$ is assumed to be a sequence of log lines solely consisting of one of these event types. The single log lines for door opening attempts have the form $l_i = (t_i, c_i, d_i)$ and the log lines for remote unlocks $l_i = (t_i, o_i, d_i)$, where $[t_1, t_2, \ldots]$ is a series of ascending timestamps, c_i a card, o_i an operator and d_i a particular door. Table 1 provides a glossary of all symbols used in this paper.

Door Sequence Model $\mathcal{D}$. The usage of every card c is captured in a weighted directed graph $\mathcal{G}_c = \langle D_c, E_c \rangle$, where the set of nodes $D_c \subseteq D$ is a fixed set of doors that have been opened using c. An edge $e_{c,d,d'} \in E_c$ reflects a transition between two doors, where d' has been opened with c after opening (and presumably passing through) d. The weight of an outgoing edge $w_{c,d,d'}$ of a door node d corresponds to the probability that the next door opening attempt of c is the target door node d'. Therefore, the weights of all edges take values within the interval [0,1] and all outgoing edges of every node sum up to 1, i.e., $\forall c \in C \ \forall d \in D_c : \sum_{d' \in D_c} w_{c,d,d'} = 1$. The timestamps of the door opening events are in ascending order and are within a time interval, which reflects the core working hours of the respective card owner. For most application cases, it is feasible to assume a break during night. Thus, we introduce an artificial end of day node $d_{eod} \in D_c$, which is not the result of a log line, but is used to 'terminate' all transitions over midnight (and start new transitions on the next day), i.e., the timestamp of the next line is on another day. As a consequence, the model skips transitions from a door d on one day to another door d' on another day. The model building approach creates separate models $\mathcal{G}_c$ for each card $c \in C$.

TABLE 1
Glossary of the Model Symbols

C	set of cards	c	card
D	set of doors	d	door
O	set of operators	o	operator
L	log data	l	log line
t	time stamp	$\mathcal{D}$	door sequence model
$\mathcal{T}$	transition time model	$\mathcal{R}$	remote unlock model
$\mathcal{G}$	directed graph	E	set of edges
e	edge	w	weight
Δ	transition time matrix	δ	entry of Δ
R	list of door unlocks	r	timestamp record
S	list of trained door sequences	s	one trained door sequence
H	list of current door sequences	h	current door sequence
P	list of last card use	p	last card use
$\mathbb{C}$	confidence value	n	sequence length in ESD
m	aggregation window length	$m_{\max}$	max. number anomalies
t_{use}	time of last usage	$t_{\max}$	max. time before removal
A	list of anomaly results		

Transition Time Model $\mathcal{T}$. The timestamps t_i of door opening events depend on the physical distances between any doors $d \in D$. Through analyzing timestamps t_i and t_j of two consecutive events reflected by l_i and l_j it is possible to determine the minimal time it takes the owner of card c to get from d_i to d_j. In the following, these times are called minimal transition times. A symmetric matrix with a diagonal of zeroes $\Delta = \{\delta_{d,d'}\}_{(d,d')\in D\times D}$ stores these minimal transition times. For easier handling of the entries of Δ, the entries $\delta_{d,d'}$ and $\delta_{d,'d}$ are interchangeable and therefore simultaneously modified. The entries $\delta_{d,d'}$ of the matrix represent the minimal transition times between two doors d and d' of any card $c \in C$, as described in Eq (1).

$$\delta_{d,d'} = \min_{(t,c,d),(t,'c,d')\in L} |t - t'| \tag{1}$$

Remote Unlock Model $\mathcal{R}$. For every door $d \in D$, our approach provides a model to capture scheduled unlocks and unlocks triggered by operators $o \in O$. The scheduled unlocks usually follow a time plan and result in remote unlock events for a specific door at precisely predefined times and week days, while manually triggered and somewhat irregular unlocks are less deterministic. The model for the remote unlocks is a list of lists $R = \{r_d\}_{d\in D}$. The lists r_d include a record of the timestamps of each remote unlock operation of door d. Note that this model does not take the operators $o \in O$ into account that trigger unlock operations. The reason for this is the rather low number of unlock operations in real use cases which would result in a lengthy training phase (see Section 5).

4.4 AMiner Detectors

Detectors of the AMiner are designed to be universally applicable and support a wide variety of log data structures. In order to demonstrate this ability, we show how to configure some detectors specifically useful for the BSS application case. Additionally, these detectors cover the use cases discussed in Section 2. The detectors described in this section all require a training phase, where the AMiner learns the detection algorithms' model reflecting the monitored system's normal behavior. Thereafter, the detectors automatically raise anomalies when log lines deviate from the learned model. Notice, this is an unsupervised approach that learns from the beginning without the use of a labeled training set.

TABLE 2
Example for Anomaly Detection Using the NCD

Card	Doors
card1	door1, door2, door3
card2	door1, door3, door4
card3	door1, door6

(a) Example for trained combinations.

Card	Doors	Anomaly
card1	door1	
card2	door3	
card3	door6	
card2	door2	x
card1	door3	
card1	door4	x

(b) Example for anomaly detection.

4.4.1 NewComboDetector (NCD)

Use cases UC#1 and UC#3 result in logged access attempts of a card c to previously never passed doors d. Most cards $c \in C$ are only used in combination with a small set of doors $d \in D$ and new combinations are added rarely to the list of benign combinations. Consequently, it is feasible to detect anomalous behavior by reporting every occurrence of a new card-door combination. The NewComboDetector (NCD) analyzes the value combinations of specific parser paths and raises anomalies when new value combinations appear.

Training Phase. For each log line $l_i = (t_i, c_i, d_i)$ processed in the training phase, any card c_i that was used to open door d_i is added to D_{c_i} of $\mathcal{G}_{c_i}$.

Anomaly Detection. In the detection phase, for every log line $l_i = (t_i, c_i, d_i)$ that does not satisfy Eq. (2) an anomaly is raised (and optionally the door d_i is added to D_{c_i} of $\mathcal{G}_{c_i}$).

$$d_i \in D_{c_i} \tag{2}$$

Example. Tables 2 a and 2 b demonstrate the detection of new card-door combinations with the NCD approach. Table 2 a includes the list of learned benign combinations. Table 2 b provides a list of combinations occuring during the detection phase. These are tested against the learned baseline, and new, i.e., anomalous combinations are marked with 'x' in the third column of Table 2 b.

4.4.2 EventSequenceDetector (ESD)

Determined by physical circumstances and steady behavior of card owners, doors are usually opened in repeating order. For instance, many card owners always enter a building through the main entrance, walk up to a particular floor where they enter a department area, and move straight into their offices. These usage patterns can be learned, and used as a baseline to detect deviations in door opening sequences.

The EventSequenceDetector (ESD) is a detector which analyzes event sequences of a given length n. The ESD monitors whether the sequence of the last n events has already been observed in the training phase and raises anomalies otherwise. The ESD can also analyze the sequence of values of a given parser path depending on the value of another path. This allows to analyze door sequences for each card c separately.

TABLE 3
Example for Anomaly Detection Using the ESD

Card	Door sequences
card1	door1→door2, door2→door3, door3→door1
card2	door1→door3, door3→door1

(a) Example for trained sequences.

Card	Door	Resulting door sequence	Anomaly
card1	door1	door3→door1	
card2	door1	door3→door1	
card2	door4	door1→door4	x
card2	door1	door4→door1	x
card1	door3	door1→door3	x
card1	door4	door3→door1	

(b) Example for anomaly detection.

Training Phase. The model $\mathcal{D}$ described in Section 4.3 is extended with a list of lists $S = \{s_c : c \in C\}$, where each list s_c refers to a card c present in the log data. Each list s_c stores all door sequences for card c that were present in the training phase. Furthermore, a second list of lists $H = \{h_c : c \in C\}$ is introduced, where each list h_c consists of the last n doors d card c was used on.

Anomaly Detection. For each new log line $l_i = (t_i, c_i, d_i)$, the ESD verifies if Eq. (3) still complies. If the sequence is of length n and not in list s_{c_i}, an anomaly is raised (and optionally the sequence added to the list).

$$h_{c_i} \in s_{c_i} \tag{3}$$

Example. Tables 3 a and 3 b show an example for the analysis of door sequences for two cards using the ESD approach. Table 3 a lists the sequences (sequence length $n = 2$) for each card $c \in C$ learned during training. Table 3 b shows the analysis of new lines. In the third column of Table 3 b anomalies are marked by 'x'.

4.4.3 TransitionTimeDetector (TTD)

Physical distances between doors result in minimal transition times in which cards can plausibly move between them. Anomalous behavior as described in UC#2 potentially results in door transitions that undercut these times. When physical distances between doors are unknown in advance during training, the minimal transition times need to be estimated.

The TransitionTimeDetector (TTD) analyzes the minimal transition times between changes of values in log lines at a given parser path. It can further be configured to distinguish events based on the values of a second path. This feature enables the analysis of minimal transition times of value changes in the first path of two log lines, while they keep the same value in the second path. Applied to our application context, this allows to track the minimal transition times between two doors (referenced in the first path) of a specific card (referenced as static value in a second path).

Training Phase. The transition time model $\mathcal{T}$ builds the basis for the TTD approach. The model is extended with a list of pairs $P = \{p_c\}_{c \in C}$ which track the transition times of cards $c \in C$. A pair $p_c = (t_c, d_c)$ includes the timestamp t_c and door d_c at which a card c was last used.

The detector checks if for card c_i of each processed log line $l_i = (t_i, c_i, d_i)$, list P contains an associated pair p_{c_i}. In that case, the transition time matrix Δ is updated. If the entry $\delta_{d_{c_i}, d_i}$ in Δ is undefined, the entry is set to the current transition time $t_i - t_{c_i}$. Otherwise, the TTD verifies if the transition time undercuts the time stated in matrix Δ as given in Eq. (4).

$$\delta_{d_{c_i}, d_i} \leq t_i - t_{c_i} \tag{4}$$

If Eq. (4) does not hold, the entry $\delta_{d_{c_i}, d_i}$ of the matrix Δ is updated with $\delta_{d_{c_i}, d_i} = t_i - t_{c_i}$. The entry p_c in P is updated with $p_c = (t_i, d_i)$ after the log line l_i has been processed.

At the end of the training phase, the minimal transition times of rare transitions are not well approximated. Therefore, after the training phase the TTD periodically checks if the transition times satisfy the triangle inequality.

The triangle inequality states that the transition time between two doors $d, d' \in D$ is not greater than the sum of the transitions of these doors to a third door $d'' \in D$ as Eq. (5) states. If an entry $\delta_{d,d'}$ is undefined and it exists a door d'', where $\delta_{d,d''}$ and $\delta_{d'',d'}$ are defined, the entry is set to the sum of these transition times. Furthermore, if the triangular inequality in Eq. (5) does not hold for any combination of doors, the entry is set to the sum of the two minimal transition times $\delta_{d,d''} + \delta_{d'',d'}$ and the process is repeated until the inequality holds for any combination.

$$\delta_{d,d'} \leq \delta_{d,d''} + \delta_{d'',d'} \tag{5}$$

Anomaly Detection. The anomaly detection works analogously to the training phase with the addition that an anomaly is raised if a transition is observed, where the minimal transition time is not defined or Eq. (6) is violated. Notice, $\epsilon \in [0, 1]$ is a threshold, introduced to lower the rate of anomalies that relate to an imprecise approximation of the minimal transition time.

The adaptation of the entries of matrix Δ can optionally be stopped to counteract the impact of anomalies on the subsequent anomaly detection. If Δ is updated, the triangle inequality is periodically verified. The confidence $\complement$ in an anomaly finding relies on the degree to which a previous transition value has been undercut. The simplest form is a linear approximation as given in Eq. (7).

$$\delta_{d_i, d_c}(1 - \epsilon) \leq t_i - t_{c_i} \tag{6}$$

$$\complement(\delta_{d_i, d_c}, t_{c_i}, t_i) = 1 - ((t_i - t_{c_i})/\delta_{d_i, d_c}) \tag{7}$$

Example. Table 4 depicts an example of the application of the TTD approach. Table 4 a depicts the minimal transition time matrix Δ which is used to analyze the transitions of the lines stated in Table 4 b. The first three columns include the values of the log lines and the other columns include the evaluation of the transition times.

4.4.4 TimeIntervalDetector (TID)

Remote unlocks (see Use Case #4 in Section 2) cannot be analyzed the same way as door accesses, because the patterns of remote unlocks are not influenced by physical distances between doors. Therefore, other analysis methods need to be applied to log lines of this event type. Most remote unlocks are scheduled and automatically issued, therefore occur at the same time of a day – with minor

TABLE 4
Example for Anomaly Detection Using the TTD

	door1	door2	door3
door1	0	01:00	02:30
door2	01:00	0	01:30
door3	02:30	01:30	0

(a) Example for trained minimal transition time matrix Δ.

Card	Door	Time	Transition time	Anomaly
card1	door1	06:00		
card2	door3	07:30		
card1	door1	08:00	02:00	
card2	door1	08:30	01:00	x ($\mathbb{C}$ = 0.6)
card1	door2	08:30	00:30	x ($\mathbb{C}$ = 0.5)
card1	door1	10:30	02:00	

(b) Example for anomaly detection.

deviations due to clock drifts or network delays. Additionally, remote unlocks with malicious intent presumably appear at times when the door is less frequently used to avert drawing attention. For these two reasons, we analyze the time intervals of remote unlocks to detect anomalous behavior.

The TimeIntervalDetector (TID) is a detector that analyzes the time intervals in which values appear in selected parser paths of log lines. Therefore, it is directly applicable to track the time intervals in which doors are remotely unlocked. Notice, we assume we do not have any knowledge of normal remote unlock behavior but learn this baseline from scratch through observation in a training phase.

Training Phase. In the training phase the timestamp t_i of each processed log line $l_i = (t_i, o_i, d_i)$ which corresponds to remote unlocks is added to list r_{d_i} of the remote unlock model $\mathcal{R}$ described in Section 4.3. In the further analysis, the TID only accounts for the week day and time of day to learn common periods when unlock operations occur.

Anomaly Detection. For each processed log line $l_i = (t_i, o_i, d_i)$ the TID checks whether the difference between the timestamp t_i and any of the timestamps $t \in r_{d_i}$ is less or equal to a parameter σ as Eq. (8) states. If no such timestamp exists, an anomaly is raised. This process results in time intervals surrounding the timestamps of the list r_{d_i} in which unlock operations occurred in the training phase and are thus allowed as depicted in Fig. 3 which is further described in the example below. Sigma can be chosen depending on how deterministic the unlock operations are performed; e.g., manual door unlocks require a larger σ than an automated system.

$$\min_{t \in r_{d_i}} |t - t_i| \leq \sigma \tag{8}$$

By default, the timestamp of the log line is added to the list r_{d_i} to adapt the model to new behavior. Optionally, this can be disabled and the model built just from a separate training phase. Lists r_{d_i} are periodically reduced, by removing timestamps which do not extend the range in which unlock operations are allowed.

Example. Fig. 3 provides an example for the analysis using the TID approach. The figure shows a timeline of 12 hours, where vertical lines symbolize one hour distances. The timestamps from list r_{d_i} are marked as circles, the resulting time

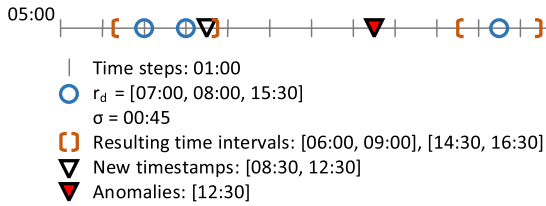


Fig. 3. Example for one normal and one anomalous timestamp detected with the TID.

intervals as brackets and the timestamps of processed log lines as triangles. While the gray triangle lies within the allowed time interval, the black triangle lies outside and results in an anomaly.

4.5 Anomaly Aggregation and Alerting

In noisy environments large quantities of anomalies may be raised. Following up on each single occurrence would easily overload security operators. Nevertheless, especially stolen or lost cards, misused by illegitimate card holders, are an underestimated problem today. Thankfully, for this use case (UC#1), we can relax our detection requirements. The assumption is that a motivated attacker systematically tests a stolen or lost card for access in short time spans, i.e., illegitimate card holders enter buildings usually once and test how far they are able to get. Therefore, we do not need to treat each single anomaly immediately, but need to detect cumulative anomalies concerning the same card in limited time spans. We apply a sliding window approach of length m individually per card c, where m represents the number of log lines from bookings of card c. If more than $m_{\max}$ log lines are reported as anomalous a in this window by, for instance, the NCD or ESD, an alert is raised. This way, we limit the sensitivity of anomaly detection, reduce the anomaly rate and put more emphasis on situations described by our use cases. Eq. (9) defines the alert model formally, where $A_c = \{a_{(c,i)} : i \in \{1, \ldots m\}\}, a_{(c,i)} \in \{0, 1\}$ is the list of anomaly results of the last m lines caused by using card $c \in C$. The value of $a_{(c,i)}$ equals one if the ith last line was anomalous and zero otherwise.

$$\sum_{i=1}^{m} a_{(c,i)} > m_{\max} \tag{9}$$

4.6 Continuous Learning and Aging

Common machine learning approaches foresee separate training and test phases to build models and test new data instances against these learned models. The introduced approach is basically not different from that, however, can be utilized also in a continuous learning mode. In this case anomalies are reported only once and the model automatically extended by the new data right afterwards. This is specifically useful in quickly changing environments, where the learned model never becomes stable. In that case, an operator could be alerted by the first occurrence of, e.g., a new card-door combination, but not by subsequent appearances.

However, with this approach, the model also continuously grows over time and will soon also contain information that does not constitute regular behavior but also exceptional cases. In an extreme case, a card will become

associated with all doors, and thus it will not be able to detect anomalies related to this card any more. We therefore foresee an important model extension if used in continuous learning mode: Every addition of a model element in $\mathcal{D}$, $\mathcal{T}$ or $\mathcal{R}$ is associated with a timestamp t_{use}, indicating its last usage (i.e., comparison with an element from a recent log line). Whenever a model element, such as a card-door combination, has not occurred in the log data for a time span $t_{\max}$, this element is removed from the model. This way, obsolete information ages out of the models. Eq. (10) defines this aging condition for removal of an element formally. Symbol t denotes the current time.

$$t - t_{\text{use}} > t_{\max} \tag{10}$$

5 Evaluation and Discussion

We demonstrate the applicability of our approach in course of an extensive real world evaluation that was carried out with data collected over a year. In order to rate the capabilities of the different models and detectors, we evaluate their detection performance separately. We investigate the impact of different model parameters, such as length of training intervals or confidence thresholds. Furthermore, we distinguish between (i) a simple anomaly detection approach, where every deviation from a trained model is considered, (ii) an approach that applies continuous learning, where deviations are reported only once and then added to the model, and (iii) an aggregated approach, where only the combined occurrence of anomalies is reported as alerts.

5.1 Data Set Description

We evaluate our concepts with an extensive data set from real world installations of the Austrian company PKE,[3] specifically from a centralized AVASYS.access management system that spans 12 physical sites of PKE across Austria and Germany. This data set contains log data that reflect the card usage over a full year in the time from March 1st, 2020 to February 28, 2021. In that time, 598364 card bookings were recorded by using 1808 unique cards at 145 distinct physical doors. A total of 11092 unique card-door combinations can be found in the data set. Notice that doors are not equally distributed over the physical sites. The largest site contains 49 doors, the second and third largest site 19 and 15 respectively. On the other side of the scale, seven locations contain less than 10 doors. This is an important fact since it explains the different card usage patterns. While some persons use their cards ten times a day, others only use them once to enter a building. Additionally, some doors need authorization in both directions, but most doors only in one direction, i.e., people need the access card to enter a building, but not to leave it.

Fig. 4 shows the number of produced log lines per day, where one log line corresponds to one booking, i.e., the usage of one card at a specific location. The figure visualizes a weekly rhythm with dips over the weekends. Notice, the first two weeks in this plot represent normal working weeks, however, after that, in mid March 2020 the first hard lock-down due to the Covid-19 pandemic caused a significant

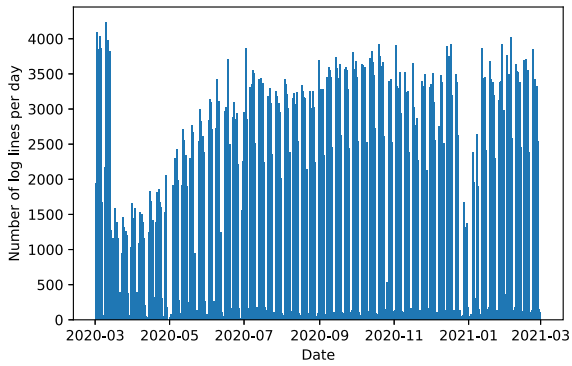


Fig. 4. Number of log lines per day.

drop in the number of daily door transitions. Similarly, around Christmas and towards the end of the x-axis of the plot a sharp drop of activity can be recognized.

Another important question is how the booking operations are distributed across the cards. As expected, a few cards are used quite extensively, while other cards are used only a few times a day at best (depending on the site of the card owner). Since our model should work without such specific domain knowledge, we provide some insights into the card usage behavior to explain the results later in this paper, but do not account for this information in the anomaly detection phase.

Fig. 5a enumerates all cards in the data set on the x-axis and shows for each card at how many *unique* doors this card has been used per day on the y-axis. The cards are sorted in descending order by the number of log lines the respective card is associated with. Additionally, the color as a third dimension provides insights into how frequently card x has been used to open y unique doors at one day and thus describe how common this usage pattern is (scaled to [0,1] whereas 1 means 100%). For instance, a card might be used at 2 unique doors most of the year (reflected by a red to orange bar), but might be used infrequently by significantly more unique doors (reflected by a blue to green bar).

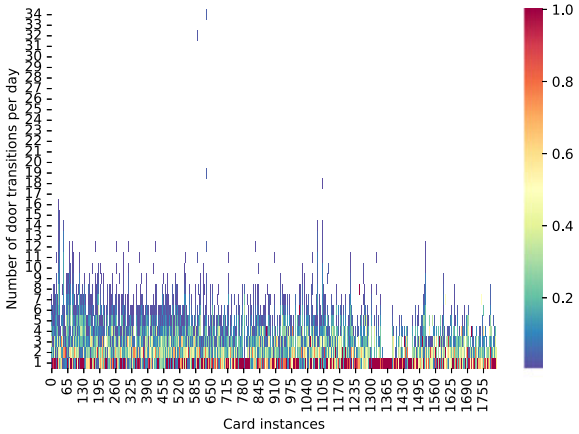
We repeat this analysis, but just for those cards that were used on at least 25% of all days in the data set (cf. Fig. 5b). This way, we filter out all rarely used cards, primarily caused by regularly executed lock-downs due to the Covid-19 pandemic. It is however noteworthy that still 719 cards (out of 1808) exhibit a somewhat 'regular' behavior, despite the frequent lock-down measures.

As for the remote unlock operations 56 doors at 9 sites are remotely unlocked by 17 different operators. From these 56 doors, 35 are located at one site alone. In total 7612 unlock operations are issued over the whole time span of the data set.

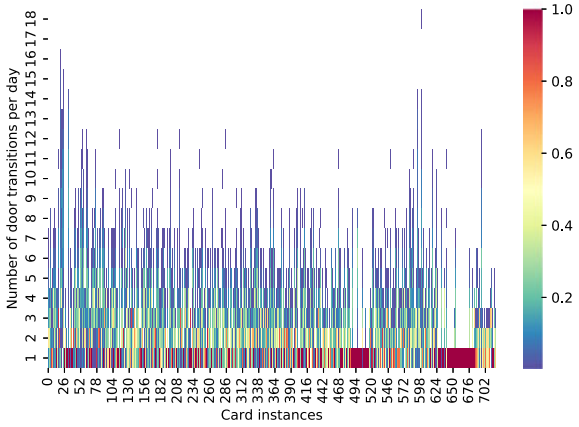
Limitations of the Data Set. Several issues are noteworthy that have impact on the further evaluation:

- *Volatile card usage:* Due to the Covid-19 pandemic and several hard lock-downs the otherwise anticipated periodic booking behavior changed dramatically. This alone constitutes numerous anomalies in the data set.
- *Temporary work assignments:* Many card owners are frequently re-assigned to different groups and departments to aid the project oriented business of PKE.

3. https://www.pke.at/



(a) for all cards.



(b) for cards that were used at least on 25% of all days.

Fig. 5. Number of used unique doors per day for each card (in descending order by the number of log lines of the respective card).

Thus, the data set is noisier, i.e., the model is less stable than expected. This would however be different in a more stable environment.

- *Changes of infrastructure:* Some door controllers were changed throughout the year which got new ids, however no mapping from the old to the new ids was performed by the system. So, several new doors emerged during the year, while older ones disappeared. This causes anomalies too with our approach.

Anomalies. During data recording, we asked some card holders to introduce anomalies. Although, we expect anomalies due to variations of daily business, this way, we had a confirmed ground truth. We knew the time stamps where our approach should detect anomalies and could use this knowledge to verify our findings. Of course, besides these known anomalies, several unknown, but likely still true positives, were detected, as discussed throughout the evaluation.

The position and properties of the introduced known anomalies are as follows:

- *Systematic Trial and Error (UC#1):* On December, first 2020, card number 490 tried to open 34 doors at one location in a time frame of 89 minutes, which should clearly be detected and stick out of the background noise.
- *Cloned Cards (UC#2):* On September, 9th 2020, card 524 was cloned and used at two different sites (Munich/DE and Vienna/AT) in a short time interval, specifically, shorter than a realistic travel time between the two sites.
- *Excessive Access Rights (UC#3):* Naturally included throughout the data set, people tend to use their cards irregularly at doors they usually do not pass through. The reason for that are quickly changing project team assignments. There was no need to artificially include such anomalies into the data set. This type of anomalies constitutes most of the noise in our data set.
- *Remote Unlock Operations (UC#4):* On December, 13th 2020, door '925 (TG)' was remotely unlocked, deliberately outside usual unlock times.

5.2 Detection of Recurrent Anomalies

In our first experiments, we directly apply the detectors as described in Section 4.4, specifically the NewComboDetector (NCD) to spot any new combinations of card-door usage and the EventSequenceDetector (ESD) to detect any breaking door transition sequences.

In this set of experiments we aim to detect *all* deviations from a trained behavior in a specified training phase. For instance, for the NCD this means that if a card c has not been used at door d in the training phase, but is then used five times after the training phase at the same door d, the system will issue five *recurrent* anomalies[4].

We first look into the application of the NCD. A key question is how many combinations that have not been observed in the training phase, the detector recognizes at all. We further like to gain more insights into the effects of different lengths of training phases on the detection results, and aim to find an appropriate length. Table 5 b shows the number of card-door combinations that were not part of the training phase. Since card usage varies tremendously, we implemented training phases per card, not for the whole system. So, we take the first 20, 50, 100, 500, 1000 and 2000 log lines of each card for training (which might take longer for rarely used cards) to build up the door sequence model $\mathcal{D}$. For each of these training phases, the number of detected anomalies in the rest of the data set is given in the respective columns. Notice, the first data column states all occurring card-door combinations without learning, thus the numbers are comparatively high.

We test the anomaly detection on data sets of different lengths. Thus, we consider only the first month, the first 3 months, the first half year or the whole year of our data set. This allows us to rate the feasibility of the training length. A careful tradeoff is required here: If the training length is too short, a high number of anomalies will be reported in the testing phase; if it is too long, it takes a considerable amount of time to create the model and enter the testing phase.

Additionally, Table 5 a reports the number of cards that produced the anomalies given in Table 5 b. For longer training phases, less cards are available in the data set that even

4. In contrast to that, *unique anomalies*, describe new card-door combinations that occurred but do not count re-occurring bookings, so only one anomaly would be reported in our simple example, instead of five.

TABLE 5
Detection Performance of the NCD for Training Lengths of 0, 20, 50, 100, 500, 1000, 2000 Log Lines Per Card

# months	0	20	50	100	500	1000	2000
01M	1165	561	261	114	2	0	0
03M	1464	899	541	299	21	2	0
06M	1622	1250	997	692	145	25	2
12M	1808	1447	1251	1053	342	162	40

(a) number of cards with sufficient data for training

# months	0	20	50	100	500	1000	2000
01M	4427	864	327	110	0	0	0
03M	5950	1776	959	486	19	1	0
06M	8169	3435	2363	1530	189	21	0
12M	11092	5706	4440	3394	1004	380	48

(b) number of unique anomalies

produced that amount of log lines. For instance, training with 100 lines is just possible for 1053 cards in the data set of one year length, and even then produces 33861 anomalies in this time span. Thus, training phases above 100 lines are barely feasible, since this would further limit the number of cards in the model.

Fig. 6 visualizes the number of NCD anomalies for each card over a time span of 12 months after training with the first 50, 100, and 200 log lines per card. Notice the changing number of cards on the x-axis, since we only consider cards with sufficient amounts of log lines in the data set and for which the training phase could be finished. Here, the number of recurrent anomalies is relevant (blue line), which is comparatively high, if they shall be treated manually (i.e., in case a security operator is being alerted each time). Nevertheless, we conclude the investigation of detecting anomalies with the NCD with the finding that increasing the training phase from 50 to 100 lines almost cuts the number of anomalies in half, however the impact of further increasing the training phase to 200 lines is much less.

Similarly to the NCD, we apply the same schema to the ESD. For the sake of brevity, we skip the detailed tables and just show the number of detected anomalies for a training phase of 100 lines per card in Fig. 7. The shape of the distribution is similar to the NCD, however, the number of detected anomalies is tremendously higher. This does not come unexpected, since the ESD detects breaking sequences and is thus much more sensitive than the NCD.

Eventually, both models for detecting recurrent anomalies are not directly applicable on our data set without further measures. Given the number of detected anomalies, it is not feasible from an economic perspective for a security operator to follow up on each instance. As a consequence of this high false positive rate, operators would likely start to ignore recurrent anomalies. We need a better model to pinpoint significant anomalies.

5.3 Detection of Unique Anomalies

Instead of detecting recurrent anomalies, in the second experiment we aim to detect unique anomalies only. This means, we enable continuous learning in our models and report only new occurrences of card-door combinations (NCD) or new occurrences of door transition sequences (ESD), and add them after the first occurrence to our models. In a real application, this would likely not happen automatically, but would require additional confirmation from a security operator to add them (and would require an aging procedure to remove them again from the model). The TransitionTimeDetector (TTD), used to detect impossible travel times between doors, and the TimeIntervalDetector (TID), used to spot unusual remote unlock operations, are applied in an unsupervised mode already, i.e., they continuously learn new values. Now, we do the same for the NCD and ESD too.

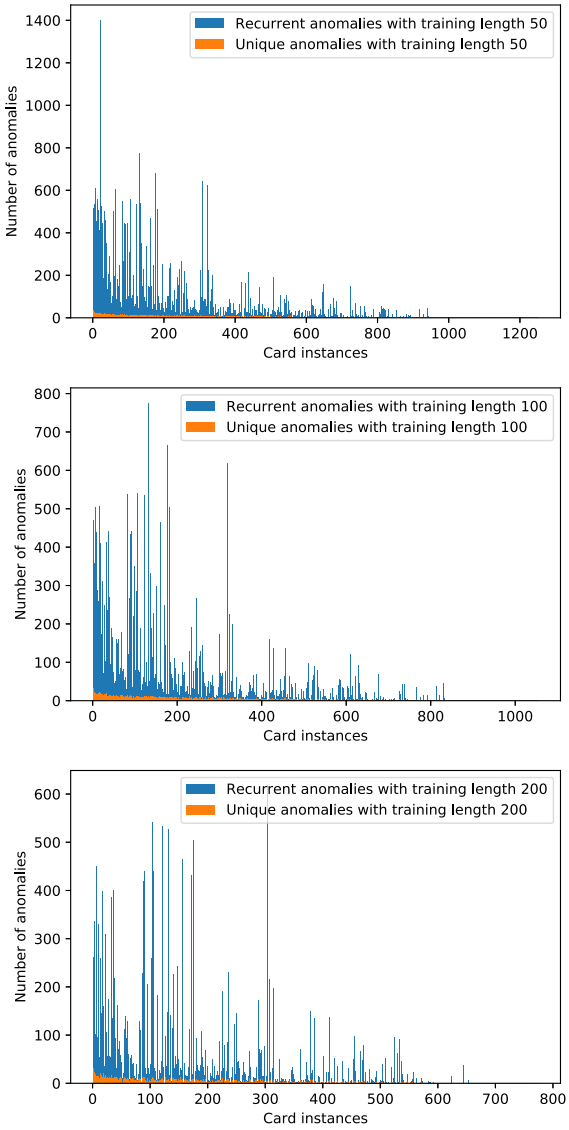


Fig. 6. NCD: Recurrent and unique anomalies.

We refer again to Fig. 6 (specifically to the configuration with training length 100) for the NCD and Fig. 7 for the ESD, however, this time we just count the unique anomalies, i.e., the first occurrence of a new combination or sequence (orange lines). As expected, the number of anomalies is greatly reduced, but anomaly reduction works much better for the NCD than for the ESD. For the NCD, only 3394 unique anomalies are reported instead of 33861 anomalies in total, which reduces the number of anomalies over the whole data set to approx. 10%. For the ESD the reduction rate is worse and the remaining amount is 24%. This does not come unexpected. Since the underlying model of the ESD is more complex and thus more prone to unique anomalies, reduction works worse than for the simpler model of the NCD.

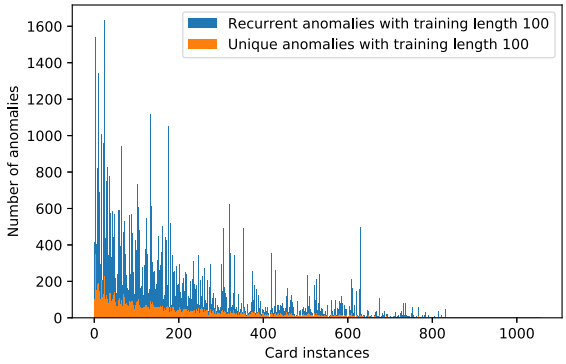


Fig. 7. ESD: Recurrent and unique anomalies.

For the NCD, this may result in a manageable number of anomalies, depending on the environment and the security requirements of the applicant. Manual investigation of the results confirmed the presence of anomalies corresponding to UC#3, i.e., staff using their excessive access rights to pass through doors that they did not use in the past. A considerable (but not otherwise defined) amount of this type of anomalies are caused by the fact that parts of the infrastructure were upgraded in the observation period, resulting in changed door ids. Nevertheless, the average anomaly rate per card for a training length of 100 lines is 3.22 (3394 unique anomalies distributed accross 1053 cards) with a standard deviation of 4.33, which is a bearable number for manual investigations in high security environment. Timewise, this would mean an investigation of approx. 9.2 anomalies per day over all cards. However, keep in mind that we investigated a quite volatile environment; thus in a high-security environment with rigid processes we would expect far smaller numbers of anomalies. We therefore deem the NCD fit for purpose to spot anomalies corresponding to UC#3 and with limitations UC#1. We relax the security requirements further and apply an anomaly aggregation mechanism in the next section to reduce the efforts of security operators.

Notice, the ESD too may detect the resulting anomalies of use cases UC#1 and UC#3. However, this detector suffers from high noise and is further not feasible for rarely used cards, where no stable sequences emerge. To counter this problem, a new door sequence h_c of a card $c \in C$ could be tested globally to appear in any sequence list $s_{c'}$ of a card $c' \in C$. This would mean that the sequence of a card usage is tested against all known sequences of all cards, and thus significantly change our use case assumptions for anomaly detection. We did therefore not follow up on that. The ESD intuitively seems to be a feasible approach for mature high-security environments with stable processes, but cannot be fully tested with our data set due to high noise.

The TTD is used to detect cloned cards, specifically their simultaneous usage at different locations (UC#2). For that purpose we ran through the full 12 months of the data set and continuously updated model $\mathcal{T}$ with the minimum times between two bookings of the same card at two distinct doors. After two weeks, we started to report anomalies. Fig. 8 visualizes the distribution and type of anomalies. The top graph shows the whole time span including the first two weeks. In the beginning the minimal transition times are only poorly estimated resulting in a high amount of

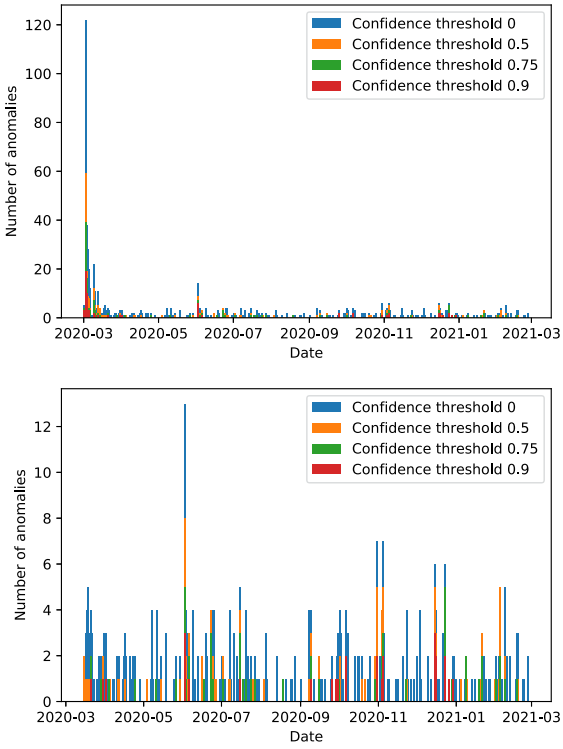


Fig. 8. TTD: Number of anomalies (top) and after a training length of two weeks (bottom).

anomalies. After two weeks, we still let the model continuously learn new minimal times, but report their occurrences as anomalies. The bottom graph shows the number of model updates, i.e., anomalies, per day. Notice, anomalies have a confidence. The more a new value undercuts the old one, the higher is the confidence in this anomaly. If we just consider the highest category (red; cutoff at 0.9 confidence) only a hand full (28) of anomalies are reported over the year. There is a significant peak in the beginning of June 2020, which is the result of newly installed doors at location DE STR, which have not been trained and therefore considered noise in the data set. Similarly, end of October to begin of November new doors were added at the location AT VIE, mid to end of December at the location AT GRZ and other single doors were added at other times that resulted in single anomalies. However, the red confidence anomaly reported beginning of September 2020 is caused by the artificially introduced anomaly as described in Section 5.1. Besides this true positive and the described noise due to infrastructure changes, the further 7 anomalies are considered false positives, which is a treatable number for one year of observation. Overall, the TTD achieves an accuracy of 0.999.[5] We could even argue that these 7 anomalies are true positives too, since we successfully detected significant infrastructure changes (that could have been the consequence of adversarial manipulations) without external knowledge. We conclude that the TTD is fit for purpose.

The TID's purpose is to detect unusual remote unlock operations at relevant doors (UC#4). Here, we distinguish between new doors that have never been unlocked, and unlock operations outside previously observed time intervals.

5. calc: 1-7[FP]/(598364[total lines]-38543[2 weeks' lines]) = 0,999987

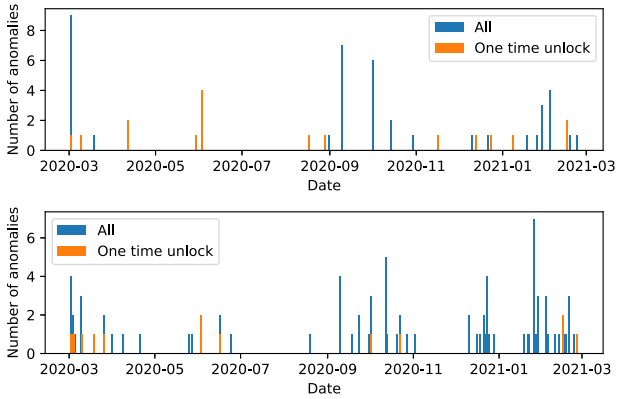


Fig. 9. TID: Number of new doors (top) and number of unlock operations outside learned intervals (bottom).

Additionally, there are two different forms of unlock operations: (i) permanent unlocks that allow repeatedly opening a door until locked again, and (ii) one time unlocks which allow exactly one person to pass through. In total we observe 7020 unlock operations, of which 592 are one time unlocks in the whole dataset over 12 months. For UC#4 we deem one time unlocks more relevant, since an attacker would need to issue only a single command and thus the hurdle is lower. Out of the 7020 unlock operations, the TID detects 56 anomalous new door unlocks (of which only 17 are one time unlocks). Likewise, out of the 7020 unlock operations, 90 of them occurred outside learned hours (respectively 14 of them were one time unlocks). Fig. 9 visualizes these numbers on a timeline. If we again apply a training period of 2 weeks, we can further decrease these numbers. Then only 15 anomalous new door unlocks and 10 unlocks outside learned intervals occurred. These add up to 25 anomalies of which one was the deliberately introduced anomaly on December, 13th 2020. The other 24 anomalies seem to be the result of unusual behavior which could not be explained as either malicious or good retrospectively. If they are treated as false positives the TID achieves a accuracy of 0.997. For an observation period of 12 months, these numbers, summarized again in Table 6, are perfectly manageable for a security operator, and we therefore consider the TID effective.

5.4 Detection of Alerts With NewComboDetector

The anomaly rate emitted by the NCD is barely economically manageable, yet this detector is crucial to cover UC#1 and partly UC#3. We therefore apply anomaly aggregation and alerting as described in Section 4 to reduce the anomaly rate. Table 7 shows the resulting number of alerts for various training lengths of 0, 20, 40 and 80 log lines per card (as indicated in the first row of the table). We tested with different window sizes. The values x/y in the first column express

TABLE 6
TID: Number of Anomalies Out of 7612 Unlock Operations

Training length	0 weeks	2 weeks
New door unlocks (total)	56	46
Out of time unlocks (total)	90	78
New door unlocks (single)	17	15
Out of time unlocks (single)	14	10

TABLE 7
Number of Created Alerts for Varying Window Lengths and for Training Lengths of 0, 20, 40 and 80 log Lines per Card

window / train.	0	20	40	80
10/2	24/2452	21/434	21/331	21/224
20/5	**17/575**	17/75	17/68	17/49
30/7	15/243	15/40	15/38	15/27
40/10	12/53	12/19	12/18	12/12
60/15	7/14	7/10	7/8	7/7
80/20	2/5	2/2	2/2	2/2

The first number represents the true positives, the second one the total number of alerts. Notice, these are massively reduced by training.

that the window size is x and y anomalies are allowed before an alert is raised. For instance, if we perform no training at all, and apply a window size of 20 log lines of which 5 anomalies are allowed before raising an alert (which effectively means that more than 5 different doors out of 20 are used by the same card without training), we receive a maximum of 575 alerts over a whole year. From these 575 alerts only 17 are true positives stemming from use case UC#1 introduced in Section 5.1. Fig. 10 depicts exactly that case. Even without training, the number of anomalies due to newly occurred card-door combinations within the specified sliding window, leading to higher number of alerts, is manageable. The most anomalous card 490 is associated with 17 alerts (positioned on the very left of Fig. 10, which is exactly the card used to demonstrate UC#1. If we additionally apply training of only 20 lines per card, we can reduce the total number of alerts over a year from 575 to just 75 of which 17 are true positives. This results in a accuracy of 5648/5706=0.99. With anomaly aggregation, we were able to isolate the anomalous card successfully.

5.5 Model Stability and Aging

Enabling continuous learning allows, optionally together with anomaly aggregation, an economically feasible detection in unsteady environments. Whether continuous learning should be used instead of a confined training phase depends on the volatility of the observed environment. Nevertheless, with continuous learning we run into the risk of extending the model with information from exceptions that are not representative for the environment. For instance, the NCD learns every new card-door combination, even when a door is used just once with a card. We therefore investigate the stability of the learned model and how quickly it ages. For that purpose, we take a closer look into the NCD and use again different training lengths of 50, 100, and 200 log lines per card. In the training phase, our approach associates cards with doors and record all card-door combinations. After the training phase is complete, we measure for each card how many doors have been learned but are not used any more in the test phase. Specifically, we divide the log data in the test phase for each card individually into batches of 40 log lines and measure the percentile of deprecated cards. Fig. 11 depicts the results. Notice, the results are smaller for shorter training lengths, since fewer cards are part of the model. While short training lengths are favorable from an aging perspective, they are more prone to high anomaly rates, e.g., caused by false positives, as shown in Fig. 6. Furthermore,

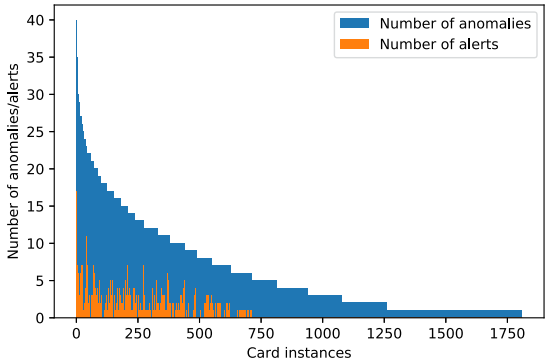


Fig. 10. NCD: Number of alerts (window length = 20, alert when more than 5 anomalies occure) and training length 0.

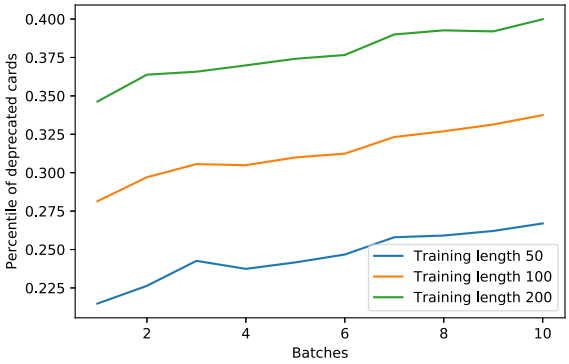


Fig. 11. NCD: Number of learned cards that did not appear in the log line batch (10 batches with 40 log lines each).

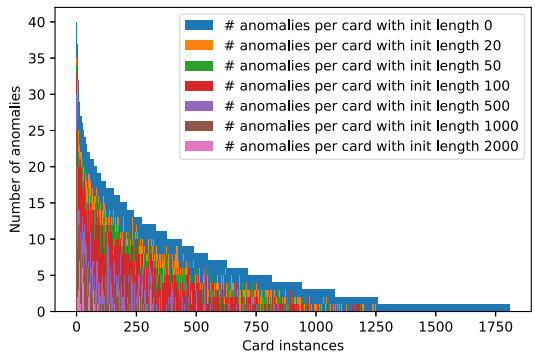
the growth rate is notable, which over all cards seems to be linear and independent from the training length. The configuration and application of an appropriate aging mechanism to remove outdated model parts seems advisable but is left for future work here.

The stability of the trained model heavily depends on the length of training. Applying a continuous learning mode, no dedicated training phase exists. Here the main question is after how many bookings (the so called 'init phase',[6] the card usage behavior is expected to be stable, and new (unique) card-door combinations shall be reported as anomalies. Fig. 12 aims to to answer this question. It shows the number of unique anomalies per card over the whole data set (i.e., doors used the first time with a specific card). With an init length of zero, the number on the y-axis corresponds to the total amount of associated unique doors with a card (shown on the x-axis). The model grows quickly in the beginning as cards are used at new doors and thus the anomaly rate drops when extending the init length. The number of anomalies dependent on the init length is a valuable metric for the model stability. Taking a closer look into Fig. 12b, where we just considered frequently used cards (on average at least once a day over the year, except Sundays), it reveals that after 100 log lines (red line) approximately half of all doors used throughout the whole year (blue line) per card, have been observed. After 500 log lines per card (purple line) 80-90% of all card-door combinations have occurred. Conversely, this means that an aging period, i.e, the time span after which model elements should be removed to avoid overtraining, should be adapted to these numbers. An NCD aging of, e.g., 100 lines, would mean that every card-door combination that has not been observed for 100 log lines, is removed from the model and subsequently alerted as anomalous. The shorter the aging period is, the higher is the potential false positive rate; the longer the aging period is, the higher is the chance for false negatives.

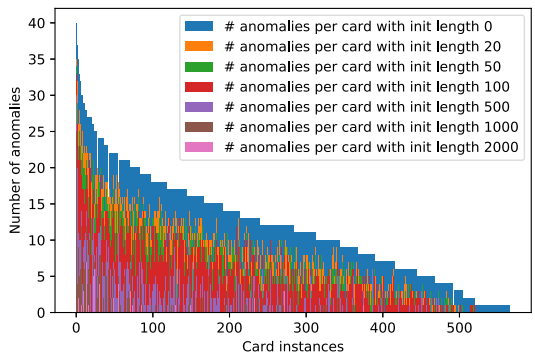
5.6 Online Anomaly Detection

Since one of our design goals was online anomaly detection, we finally investigate how quickly we are able to detect malicious behavior. While this is comparatively easy to achieve for TID and TTD, since here a single value deviating from the model may immediately cause an alert by design, it is much more challenging for the NCD and ESD, where from potentially thousands of anomalies the significant ones need to be isolated in a timely manner. For our measurements, we pick again the NCD and apply aggregation of unique anomalies as outlined before. Table 8 shows the time delays in thousands of seconds from the first detected anomaly to an aggregated alert. An alert is raised if m_t anomalies occurred in the last m events. At first sight, regardless from the configuration, the time delays seem to be extremely high, potentially hundreds of hours. However, this is perfectly fine if we consider anomalous bookings over longer time spans, potentially days, where card holders might try to gradually evaluate the limits of their access rights.

Looking into the standard deviation in Table 8 already reveals that the distribution of time delays is key for the



(a) NCD: Unique anomalies per card.



(b) NCD: Unique Anomalies per card for cards used more than 300 times in the whole data set.

Fig. 12. NCD: Unique Anomalies with different init lengths.

6. Notice, we carefully distinguish the init phase from the training phase, since after the init phase the model is still extended with every new card-door combination, while after a training phase, the model is frozen.

TABLE 8
NCD: Mean and Median of Time Delays in Thousands of Seconds From the First Detected Anomaly to an Aggregated Alert

	all cards			cards with $>$ 100 log lines		
m/m_t	mean	median	std dev.	mean	median	std dev.
10/2	1017	81	2907	462	57	1377
20/5	1999	405	3877	1189	261	2312
30/7	2847	777	5055	1944	451	3846
40/10	3308	277	6361	2031	172	4255
60/15	8142	3119	8925	8142	3119	8925
80/20	13660	21269	11189	13660	21269	11189

evaluation of alerts. Fig. 13 visualizes these delays. The alerts with a rather small delay on the left side (around 10^{-2} days corresponding to approx. 8.6 to 28.8 minutes) are the result of the introduced anomaly (cf. Section 5.1), while the anomalies to the right are considered false positives. The true positives are the result of someone systematically trying out new doors (i.e., one door every few minutes), while the anomalies on the right side are the result of accidentally used new doors over longer time spans. These numbers also match the true positives outlined in Table 7 where 7 out of 14 alerts are considered true positives. Eventually, distinguishing aggregated alerts due to their time delay has huge benefit to sort out false positives and timely, i.e., within minutes, revealing the anomalies stemming from the defined use cases.

6 Lessons Learned

Various lessons were learned in the deployment and evaluation of the detectors. We discuss specific findings on the suitability of the detectors as well as general thoughts on the applicability of AD for the given use cases. The findings are substantially based on the measured detection accuracy in course of the evaluation.

6.1 Limitations of the Data Set

The data set suffers from some limitations, most noteworthy, changing user behavior due to the pandemic, as well as changes in the infrastructure (door controllers were replaced). We carefully designed the detectors, in consultation with the system owner, to make them resilient against these issues.

6.2 Initial Deployment of Detectors

For the evaluation, we varied the detector configurations for a wide range of parameters, e.g., initialization lengths, training lengths and confidence thresholds. The resulting set of curves are discussed in detail in Section 5. These figures are supposed to reveal the impact of the different configurations on the final results, however, are not generally valid, but specifically applicable to the investigated data set. The suitability of the parameters rely on the detection use cases (and anomalies to be discovered), the number of doors per site, the number of users on these sites and the average door transitions per card. Parameters need to be set carefully in order to obtain useful results, i.e., reliably discover anomalies due to misuse, and on the other side not to suffer from noise (i.e, anomalies due to legitimate behavior variations). One aspect, which is hard to estimate, is the stability of usage behavior of

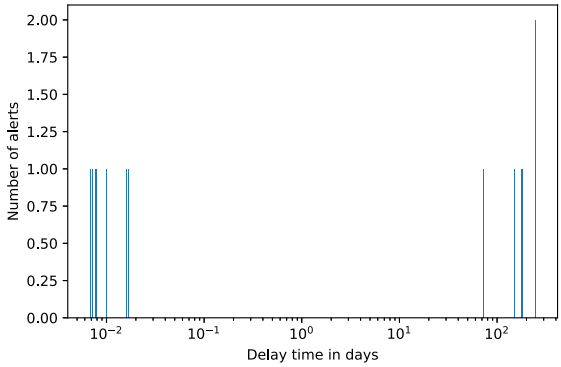


Fig. 13. NCD: Histogram of the time delays in days from the first detected anomaly to an alert ($m = 60, m_t = 15$ training length = 0).

card owners, i.e., how deterministic is the booking behavior of people without major interruptions through holidays and inconsistencies through changes of work assignments. Stability was most notably assumed in the detector design of the ESD, but also to a smaller extent in the NCD. Both detectors have great potential to successfully reveal even subtle anomalies, however, they are prone to varying user behavior. Eventually, for a successful configuration of the detectors, the stability of user behavior must be verified at first before setting aforementioned parameters.

6.3 Continuous Improvements

After the initial deployment our system delivered poor performance, which lead to an unacceptably low accuracy, e.g., the NCD had an accuracy of 0.943 with a training length of 100 lines.

Thus, we continuously tested and adapted the applied detectors. We performed parameter selection analysis and performance analysis, which lead to the concept of anomaly aggregation (see Section 5.4) into alerts. Furthermore the continuous learning and aging process of the behavior models was investigated (see Section 5.5). We discuss our lessons learned regarding these aspects in the following.

6.3.1 *Parameter Selection*

The optimal parameter selection of the training length depends on the specific detector and data set properties. Both the TTD and TID were easily configured by applying a training length of two weeks, because of their user-independent model design. The model designs of the NCD and ESD analyze the cards individually and therefore the user based training length always results in a trade off between the initialized model stability and the number of analyzed cards in a given time window. The analysis yielded that a training length higher than 100 was not feasible due to excluding too many cards from the analysis.

Another investigated parameter is the confidence threshold of the TTD, which states with which factor the transition time has to be undercut to raise an alert. The evaluation yielded that the purposefully introduced anomaly was found with a threshold of 0.9, while simultaneously reducing the number of false positives from 324 to 7. The threshold could be reduced to discover more subtle anomalies, but was sufficient to detect the introduced ones. Notice, the

optimal threshold depends on the distance between sites and can be set lower if they are situated closer to each other.

6.3.2 Performance Analysis

While the TTD and TID yielded sufficiently good results (i.e., we detected 100% percent of the synthetically added anomalies) with an economically small number of false positives (i.e., 7 and 24 anomalies for TTD and TID respectively, which corresponds to maximal two anomalies per month and detector), both the NCD and ESD reported too many anomalies (NCD 3394 and ESD 20878 anomalies with training length of 100 lines per card). This is a result of the assumed stability of the user behavior in the model design, which was however not present in the analyzed log data. The main problem was the temporary work assignments of the card users, which resulted in ever changing card door combinations and sequences.

To reduce the number of anomalies the aggregation into alerts was introduced. While it was not possible to aggregate the anomalies of the ESD into informative alerts,[7] the NCD reported a sufficiently small number of alerts after aggregation, i.e., a drop from 5706 anomalies to 75 alerts (of which 17 are TPs) with a time window of 20/5 and trainings length of 20, while maintaining the detection of the purposefully introduced anomaly.

6.3.3 Aging

While the continuous learning of the detectors yielded an effective reduction of reoccurring false positives, it also introduced the problem of an ever growing model. The aging method was introduced to simultaneously reduce the complexity of models and enhancing the anomaly detection by removing old information. Fig. 11 shows that approx. 5% of unused doors drop out per batch (each 40 log lines per card) when progressing along the timeline.

6.4 Result Interpretation

Unfortunately, anomaly detection results are not easy to interpret. The system reports any deviation from the learned baseline, the art however is to distinguish deviations due to malicious behavior from deviations due to unsteady (but natural) behavior of legitimate card holders.

6.4.1 Anomaly/alarm Arrival Rate

With all described modifications a total number of 89 alerts were falsely raised in the analysis of the whole data set. The NCD, TTD and TID raised 58, 7 and 24 false alerts respectively. If the trainings length is assumed to be two weeks long, this corresponds to about one false alerts every four days, which appears manageable. The number of false alerts scales directly with the number of cards and their respective usage behavior or remote unlock operations.

6.4.2 Application of the Classic FPR Metric

The low number of actual positive population in comparison to the actual negative population is the reason why the F-score was not suitable for the evaluation [17]. The accuracy[8] is an easily comprehensible measure, which states the percentage of correct classification of lines into their associated population. This means, that the accuracy equally weights the actual positive and negative population. We argue that the detectors are reasonably precise, because they detected all synthetically injected anomalies, and therefore the possible distortion of the accuracy due to class imbalance is negligible. The accuracy of the NCD, TTD and TID is 0.99, 0.999 and 0.997 respectively, which are reasonably good results, i.e., the number of expected anomalies can be handled by a security operator as outlined above.

7 Background and Related Work

The topic of this paper involves a number of different technical aspects. We structure the review of related work therefore into (i) monitoring and logging, (ii) intrusion detection systems, (iii) anomaly detection, and (iv) application in cyber-physical systems, specifically in building management/security systems.

7.1 Log Data

Data logging has a widespread application area in information and communication technology. Log data is an important source for system monitoring [18], which comprises acquisition of data and knowledge. Furthermore, log data is investigated in course of digital forensics [19], which is applied, for example, after an attack was detected to investigate its origin and find out information about the attacker and the purpose of the attack. Database logs can be used to back up and restore database content in case of a system crash or a destruction caused by an unauthorized access violation [20], [21], [22]. Log data contains automatically generated traces about all processes of services and components of a computer network. Thus, it protocols all events occurring in such networks. Log data is usually represented in human-readable text format. This makes it easy to access the provided information. Other data sources, such as network packets, require time-, computational- and resource-intensive preprocessing before analysis. Thus, log data is a valuable source for cyber security analysis tools, such as IDS. The level of detail of information provided by log data depends on the configuration of the logging mechanisms [23]. One drawback is that usually only logs of lower severity levels, i.e., warning and error logs, are stored and used for security analysis. But to carry out extensive analysis, verbose logging is required, which, however, often is not the case. Reasons are that logging often is a resource consuming task and produces large amounts of data that have to be stored [1].

7.2 Anomaly Detection

The rapidly changing cyber threat landscape requires flexible, self-learning and adaptive IDS approaches. One solution is self-learning anomaly detection (AD) [24]. They usually learn during a training phase a baseline of normal system behavior that then serves as ground truth to detect anomalies that expose attack traces. Generally, there are three ways how self-learning AD can be realized [7], [10],

7. There were too many alerts: 1038 alerts (of which 17 are TPs) with a long window of size 80/20 and training length of 100

8. Accuracy = (TP + TN) / (TP + FP + TN + FN)

[25]: (i) *Unsupervised:* This method does not require any labeled data and is able to learn to distinguish normal from malicious system behavior during the training phase. Based on the findings, it classifies any other given data during the detection phase. (iii) *Semi-supervised:* This method is applied when the training set only contains anomaly-free data and is therefore also called 'one-class' classification. (iii) *Supervised:* This method requires a fully labeled training set containing both normal and malicious data. These three methods do not require active human intervention during the learning process. While unsupervised self-learning is entirely independent from human influence, for the other two methods the user needs to ensure that training data is anomaly free or correctly labeled. Consequently, the previously described methods are categorized as unsupported self-learning approaches [7], [25].

Using completely unsupported self-learning raises some challenges. While providing training data for unsupervised self-learning is rather easy and does not require any preprocessing, this approach might learn malicious system behavior as normal. Semi-supervised approaches try to avoid this problem by using anomaly-free training data. Applying these methods raises the problem of obtaining training data that guarantees to be anomaly-free. Retrieving such a dataset from a running productive system is usually difficult, because organizations are often not aware of malicious activities in their computer network. Also for self-learning based anomaly detection it is true that the more information is provided during the training phase, the more accurate the system works later while a computer network is monitored. Thus, supervised self-learning approaches provide the most detailed ground truth, but providing suitable training data for specific networks is time- and resource consuming.

To avoid the mentioned drawbacks, supported self-learning approaches can be applied, where also system administrators can influence the training phase. This means for example, when an event is occurring for the first time and therefore is not part of the normal system behavior and as a consequence is classified anomalous, the administrator can decide if the event comprises an anomaly or if it is a false alarm and the event should be considered as normal system behavior in the future. Furthermore, self-learning can be used to constantly adapt the baseline, which describes the normal behavior and keep it up-to-date, when, for example, new devices are added to or removed from a network, or when the used software changes because of system updates.

Aside from all the advantages of AD based detection methods, they have also been shown to be vulnerable to a certain type of attack [26]: When using carefully manipulated input samples, i.e., adversarial examples, a threat actor can circumvent detection or cause erroneous predictions (high-confidence misclassification). AD based methods are especially affected by poisoning attacks, where attackers try to manipulate the training phase in a way that during the detection phase their attacks are accepted as benign system/network behavior [27].

7.3 Machine Learning

There exist many machine learning [28] algorithms to implement self-learning AD that can be applied for that task. Methods that are used for machine learning in cyber security are, for example [25], [29]: (i) *Clustering:* Clustering enables grouping of unlabeled data. It is often applied to detect outliers and forms the foundation for generating log parsers that define a system's normal behavior [13], [30], [31]. (ii) *Artificial neural networks (ANN):* Input data activates neurons (nodes) of an artificial network, inspired by the human brain. The nodes of the first layer pass their output to the nodes of the next layer, until the output of the last layer of the artificial network classifies the monitored computer networks' current state [32]. (iii) *Bayesian networks:* Bayesian networks define graphical models that encode the probabilistic relationships between variables of interest and can predict consequences of actions [33]. (iv) *Decision trees:* Decision trees have a tree-like structure, which comprises paths that lead to a classification based on the values of different features [34]. (v) *Hidden markov models (HMM):* A Markov chain connects states through transition probabilities. HMM aim at determining hidden (unobservable) parameters from observed parameters [35]. (vi) *Support vector machines (SVM):* SVM construct hyperplanes in a high- or infinite-dimensional space, which then can be used for classification and regression. Thus, similar to clustering, SVM can, for example, be applied for outlier detection [36].

7.4 Security of Cyber-Physical Systems

Numerous research works addressed the different security aspects of the vast field of Cyber-Physical Systems (CPS), including production systems [37], smart cities, smart energy grids [4], medical application areas and so on. The various CPS security goals were discussed in [38]; CPS security challenges and issues were presented, including Big Data security [39], and IoT issues [40]; several security and privacy solutions using cryptographic mechanisms were discussed in [41]. Building management systems [42] utilize one form of cyber physical systems. IoT based smart security for home automation systems has been investigated by [43], while [44] reviewed smart home security in detail. Our work specifically focuses on physical access control, as applied by building security systems (BSS) as part of larger building management systems (BMS) [45].

Physical access control has been investigated in more detail by [6], who report on the core set of features a physical access control system should have baed on examining existing systems, industry standards and government regulations. Some works, such as [46], already introduced approaches to detect suspicious patterns in physical access control. However, their main purpose is different from ours. While our approach was designed for self-learning a baseline and subsequent online anomaly detection (of deviating behavior), common approaches were designed for the discovery of door unlock sequences which are fundamentally impossible by checking them against a defined plan (and not against observations over time as our approach). The concept of "expected behavior" is therefore fundamentally different from our works, since theirs is pre-defined by a plan, while ours is created (and continuously adapted) through observations.

8 Conclusion and Future Work

In this paper, we investigated the use of anomaly detection in log data of cyber-physical systems. We outlined generally

applicable detection approaches and demonstrated their capabilities on a real-world data set from a building security system. There, we studied detection capabilities in context of different use cases in the domain. The experiments showed that with careful configuration of training phases and the use of aggregation features, our system is able to find misuse stemming from stolen or cloned cards.

We introduced four different detectors which covered varying parts of the initially presented use cases and which performed with varying degree of success. It is noteworthy that the design goal of our anomaly detection approach, applied in building security systems, was not to reveal every single unusual booking or door traversal, which is technically possible but results in unbearably high numbers of anomalies. Instead, our goal was to detect strong changes of the underlying behavior of e.g., card holders and their booking behavior. This significantly relaxes the otherwise tight requirements on anomaly detection, since aggregation and longer observation periods aid the targeted detection of such behavior shifts and reduces the noise and potentially the number of false positives. In an extensive evaluation with a real data set, we demonstrated the detection performance and its dependency on the learning mode, training length, and use of anomaly aggregation. Eventually, we showed that with optimized settings of the machine learning detectors, we were able to discover traces of misuse within the defined use cases.

Future works is manifold. The consideration of changes and updates in the environment, e.g., the replacement of door controllers or the re-assignment of access cards to new personnel is key to increase the detection performance. Such changes should trigger a new learning phase or targeted purge of the learned model for the concerned cards or doors. Further research is required concerning the update strategy of trained models in conjunction with the discussed aging approach. While the continuous learning mode is useful in volatile environments, where no steady behavior models emerge, it is also tricky to configure an underlying aging approach appropriately. There is currently no strategy to come up with feasible configuration parameters of such models.

Another important extension involves the consideration of contextual elements, such as the operational role of a card holder, and information on project assignments or leaves. We expect that accounting for such information can increase the anomaly detection performance respectively the interpretation of anomalies by distinguishing expected behavior deviations from unexpected ones. Eventually, the case study of this paper provides interesting insights into the applicability of anomaly detection for security purposes in cyber-physical systems.

REFERENCES

[1] A. Chuvakin, K. Schmidt, and C. Phillips, *Logging and Log Management: The Authoritative Guide to Understanding the Concepts Surrounding Logging and Log Management*, Oxford, United Kingdom: Newnes, 2012.

[2] F. Skopik, M. Wurzenberger, and M. Landauer, *Smart Log Data Analytics: Techniques for Advanced Security Analysis*, Berlin, Germany: Springer, 2021.

[3] S. Huda et al., "Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data," *Inf. Sci.*, vol. 379, pp. 211–228, 2017.

[4] F. Skopik and P. Smith, *Smart Grid Security: Innovative Solutions for a Modernized Grid*, Oxford, United Kingdom: Syngress, 2015.

[5] M. Shashanka, M.-Y. Shen, and J. Wang, "User and entity behavior analytics for enterprise security," in *Proc. IEEE Int. Conf. Big Data*, 2016, pp. 1867–1874.

[6] E. B. Fernandez, J. Ballesteros, A. C. Desouza-Doucet, and M. M. Larrondo-Petrie, "Security patterns for physical access control systems," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*, 2007, pp. 259–274.

[7] M. Wurzenberger, F. Skopik, G. Settanni, and R. Fiedler, "AECID: A self-learning anomaly detection approach based on light-weight log parser models," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 386–397.

[8] M. E. Whitman and H. J. Mattord, *Principles of Information Security*, 6th ed. Boston, MA, USA: Cengage Learning, 2017.

[9] H.-J. Liao, C.-H. Richard Lin, Y.-C. Lin, and K.-Y. Tung, "Intrusion detection system: A comprehensive review," *J. Netw. Comput. Appl.*, vol. 36, no. 1, pp. 16–24, Jan. 2013.

[10] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS One*, vol. 11, no. 4, 2016, p. e0152173.

[11] T. Chen, L.-A. Tang, Y. Sun, Z. Chen, and K. Zhang, "Entity embedding-based anomaly detection for heterogeneous categorical events," 2016, *arXiv:1608.07502*.

[12] A. Taha and A. S. Hadi, "Anomaly detection methods for categorical data: A review," *ACM CSUR*, vol. 52, no. 2, pp. 1–35, 2019.

[13] M. Landauer, F. Skopik, M. Wurzenberger, and A. Rauber, "System log clustering approaches for cyber security applications: A survey," *Comput. Secur.*, vol. 92, 2020, Art. no. 101739.

[14] V. K. Rohatgi and A. M. E. Saleh, *An Introduction to Probability and Statistics*. Hoboken, NJ, USA: Wiley, 2015.

[15] R. Balakrishnan and K. Ranganathan, *A Textbook of Graph Theory*, Berlin, Germany: Springer, 2012.

[16] M. Wurzenberger, M. Landauer, F. Skopik, and W. Kastner, "AECID-PG: A tree-based log parser generator to enable log analysis," in *Proc. IFIP/IEEE Symp. Integr. Netw. Serv. Manage.*, 2019, pp. 7–12.

[17] C. K. Williams, "The effect of class imbalance on precision-recall curves," *Neural Comput.*, vol. 33, no. 4, pp. 853–857, 2021.

[18] S. E. Hansen and E. T. Atkins, "Automated system monitoring and notification with swatch," *Library Inf. Sci. Abstr.*, vol. 93, 1993, pp. 145–152.

[19] S. Raghavan, "Digital forensic research: Current state of the art," *CSI Trans. ICT*, vol. 1, no. 1, pp. 91–114, 2013.

[20] R. Fang, H.-I. Hsiao, B. He, C. Mohan, and Y. Wang, "High performance database logging using storage class memory," in *Proc. IEEE Int. Conf. Data Eng.*, 2011, pp. 1221–1231.

[21] P. Frühwirt, P. Kieseberg, S. Schrittwieser, M. Huber, and E. Weippl, "Innodb database forensics: Reconstructing data manipulation queries from redo logs," in *Proc. IEEE Int. Conf. Availability Rel. Secur.*, 2012, pp. 625–633.

[22] M. Wurzenberger, F. Skopik, G. Settanni, and W. Scherrer, "Complex log file synthesis for rapid sandbox-benchmarking of security-and computer network analysis tools," *Inf. Syst.*, vol. 60, pp. 13–33, 2016.

[23] R. Gerhards, "The syslog protocol: Rfc 5424," *Internet Eng. Task Force*, 2009.

[24] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[25] M. Wurzenberger, F. Skopik, and G. Settanni, "Big data for cybersecurity," in *Encyclopedia of Big Data Technologies*, S. Sakr and A. Zomaya, Eds, Berlin, Germany: Springer, 2018, pp. 1–9.

[26] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[27] M. Kloft and P. Laskov, "Online anomaly detection under adversarial impact," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 405–412.

[28] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical Machine Learning Tools and Techniques*, San Mateo, CA, USA: Morgan Kaufmann, 2016.

[29] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Comm. Surv. Tut.*, vol. 18, no. 2, pp. 1153–1176, Apr.–Jun. 2016.

[30] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Trans. Neural Netw.*, vol. 16, no. 3, pp. 645–678, May 2005.

[31] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, 2016, pp. 654–661.

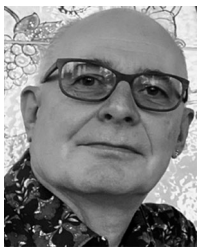
[32] J. Cannady, "Artificial neural networks for misuse detection," in *Proc. Nat. Inf. Syst. Secur. Conf.*, 1998, pp. 443–456.
[33] D. Heckerman et al., "A tutorial on learning with bayesian networks," *Nato Asi Ser. D. Behav. Social Sci.*, vol. 89, pp. 301–354, 1998.
[34] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.
[35] L. E. Baum and J. A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," *Bull. Amer. Math. Soc.*, vol. 73, no. 3, pp. 360–363, 1967.
[36] I. Steinwart and A. Christmann, *Support Vector Machines*, Berlin, Germany: Springer, 2008.
[37] G. Settanni, F. Skopik, A. Karaj, M. Wurzenberger, and R. Fiedler, "Protecting cyber physical production systems using anomaly detection to enable self-adaptation," in *Proc. IEEE Ind. Cyber-Phys. Syst.*, 2018, pp. 173–180.
[38] E. Bou-Harb, "A brief survey of security approaches for cyber-physical systems," in *Proc. IEEE 8th IFIP Int. Conf. New Technol. Mobility Secur.*, 2016, pp. 1–5.
[39] H. Ye, X. Cheng, M. Yuan, L. Xu, J. Gao, and C. Cheng, "A survey of security and privacy in Big Data," in *Proc. Int. Symp. Comm. Inf. Techn.*, 2016, pp. 268–272.
[40] J. S. Kumar and D. R. Patel, "A survey on Internet of Things: Security and privacy issues," *Int. J. Comput. Appl.*, vol. 90, no. 11, pp. 20–26, 2014.
[41] O. Kocabas, T. Soyata, and M. K. Aktas, "Emerging security mechanisms for medical cyber physical systems," *IEEE/ACM Trans. Comp. Biol./Bioinf.*, vol. 13, no. 3, pp. 401–416, May/Jun. 2016.
[42] D. Minoli, K. Sohraby, and B. Occhiogrosso, "IoT considerations, requirements, and architectures for smart buildings–energy optimization and next-generation building management systems," *IEEE Internet Things J.*, vol. 4, no. 1, pp. 269–283, Feb. 2017.
[43] R. K. Kodali, V. Jain, S. Bose, and L. Boppana, "IoT based smart security and home automation system," in *Proc. IEEE Int. Conf. Comput. Commun. Autom.*, 2016, pp. 1286–1289.
[44] R. J. Robles and T.-H. Kim, "A review on security in smart home development," *Int. J. Adv. Sci. Technol.*, vol. 15, pp. 13–22, 2010.
[45] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang, "Securing building management systems using named data networking," *IEEE Netw.*, vol. 28, no. 3, pp. 50–56, 2014.
[46] S. Fong and Z. Yan, "A security model for detecting suspicious patterns in physical environment," in *Proc. IEEE 3rd Int. Symp. Inf. Assurance Secur.*, 2007, pp. 221–226.

Florian Skopik (Senior Member, IEEE) received the PhD degree in computer science from the Vienna University of Technology. He is head of the cybersecurity research program with the Austrian Institute of Technology. His main interests are centered on critical infrastructure protection and intrusion detection. He is a member of various conference program committees (e.g., ACM SAC, ARES, CRITIS), editorial boards, and standardization groups, such as ETSI TC Cyber, IFIP TC11, and OASIS CTI

Markus Wurzenberger received the PhD degree in computer science from the Vienna University of Technology. He is a scientist and project Manager with the Austrian Institute of Technology. His main research interests are log data analysis with a focus on anomaly detection (AD) and cyberthreat intelligence. He is one of the key researchers working on the AMiner, a software component for log analysis, which implements several AD algorithms and is included as package in the official Debian Linux distribution.

Georg Höld received the bachelor's degree in technical mathematics, from the the Vienna University of Technology, in 2018. He is a junior scientist at the Austrian Institute of Technology. Since then, he is part of the cyber security research program, working on novel anomaly detection methods applicable on log data. He currently pursues a Masters degree in technical mathematics with TU Vienna in collaboration with the AIT.

Max Landauer is currently working toward the PhD degree in computer science with the Vienna University of Technology, in collaboration with the Austrian Institute of Technology. He is a scientist with the Austrian Institute of Technology. His main research interests are log data analysis, anomaly detection (AD), and cyberthreat intelligence. For his dissertation, he is working on an approach that extracts actionable cyberthreat intelligence from raw log data.



Walter Kuhn is the designer of the AVASYS security management system with PKE Holding AG used to generate the data in this paper. He started his activities with the Austrian Institute of Technology as Business Unit Manager, where he also designed the previous version of the AVASYS system. The research focus was on safety-critical and fail-safe systems. After moving to PKE, he established the Research and Development department there and worked on several funded research projects.