International Journal of Information Security (2025) 24:3
https://doi.org/10.1007/s10207-024-00921-0

REGULAR CONTRIBUTION

A review of time-series analysis for cyber security analytics: from intrusion detection to attack prediction

Max Landauer[1] · Florian Skopik[1] · Branka Stojanović[2] · Andreas Flatscher[2] · Torsten Ullrich[3]

Published online: 30 October 2024

Abstract

Understanding the current threat landscape as well as timely detection of imminent attacks are primary objectives of cyber security. Through time-series modeling of security data, such as event logs, alerts, or incidents, analysts take a step towards these goals. On the one hand, extrapolating time-series to predict future occurrences of attacks and vulnerabilities is able to support decision-making and preparation against threats. On the other hand, detection of model deviations as anomalies can point to suspicious outliers and thereby disclose cyber attacks. However, since the set of available techniques for time-series analysis is just as diverse as the research domains in the area of cyber security analytics, it can be difficult for analysts to understand which approaches fit the properties of security data at hand. This paper therefore conducts a broad literature review in research domains that leverage time-series analysis for cyber security analytics, with focus on available techniques, data sets, and challenges imposed by applications or feature properties. The results of our study indicate that relevant approaches range from detective systems ingesting short-term and low-level events to models that produce long-term forecasts of high-level attack cases.

Keywords Time-series analysis · Cyber security analytics · Intrusion detection · Attack prediction · Security data sets

1 Introduction

Cyber analysts continuously monitor and assess the threat landscape. Recent reports thereby indicate a strong increase of both the number and variety of cyber attacks, with ransomware, threats against availability, and intrusions in cloud computing environments currently ranging among the most critical attack vectors [63, 69]. To alleviate these threats, security experts employ advanced analysis techniques that aim to predict the spread of such attacks and provide countermeasures for detection and mitigation.

Across all domains of cyber security, analysts have long understood the importance of this kind of security data analytics. In particular, collection, analysis, and interpretation of such data are key elements in ensuring security, for example, when it comes to live monitoring of systems and networks for the purpose of threat detection, forensic investigations of cyber incident timelines in security operation centers (SOC), or estimation and prediction of current trends in the threat landscape [110, 125].

The number and diversity of potential data sources for the collection of security data is enormous, and dealing with massive amounts of data in various formats is a challenging task [122]. A common feature of most security data sets is that they include some kind of timestamps or time-related attributes, which are essential for contextualizing the data and correlating events with each other. This time information could thereby, for example, relate to the timestamp of an event taking place on a system, the date of a cyber incident or its detection, the discovery of a vulnerability, etc.

Due to this temporal nature of most security data sets, time-series analysis (TSA) is a natural choice for analysts.

✉ Max Landauer
max.landauer@ait.ac.at

Florian Skopik
florian.skopik@ait.ac.at

Branka Stojanović
branka.stojanovic@joanneum.at

Andreas Flatscher
andreas.flatscher@joanneum.at

Torsten Ullrich
torsten.ullrich@fraunhofer.at

[1] Austrian Institute of Technology, Vienna, Austria

[2] Joanneum Research, Graz, Austria

[3] Fraunhofer Austria Research, Graz, Austria

The main benefit of TSA is that through analysis and modeling of the data that occurred in the past, one is able to forecast future trends and react to them appropriately [68]. In the context of cyber security, this can mean to project what types of attacks or attack steps will happen next, when or where they will occur, and how the overall situation is evolving, i.e., how many attacks are expected to occur in an upcoming time period [50].

TSA comprises a versatile set of methods, and many techniques can be applied in various application domains of cyber security. However, properties of the analyzed data, such as mixtures of categorical and numeric features, seasonality, or sparsity, limit the effectiveness of certain techniques with respect to the model and prediction accuracy. As a consequence, it is not straightforward to select and configure adequate TSA methods for a specific use-case at hand.

To the best of our knowledge, there is currently no extensive overview that analyzes the big picture of TSA in the cyber analytics domain. Existing surveys are either outdated [26], consider application of TSA without security context [18, 24], or focus only on specific application cases such as smart grids [105], attack prediction [51], or anomaly detection for Internet-of-Things devices [27].

With this paper we aim to address the aforementioned gaps by providing an overview of TSA in the research domain of cyber security analytics. To this end we propose the following research questions for our study:

- RQ1: Which domains of cyber security analytics involve time-series analysis?
- RQ2: What are suitable sources of security-relevant data for time-series analysis?
- RQ3: What constraints and challenges do data and use-cases impose on the application of time-series analysis and how are they overcome by approaches?

In this paper we answer these research questions through a broad review of relevant literature. Specifically, we search and select scientific publications that apply methods from TSA on data sets collected in cyber security application domains, e.g., intrusion detection and attack prediction. We point out that due to the enormous amount of publications that could potentially be considered relevant for this study, we only aim to gather a representative sample of papers for the main application domains and focus on the overall trends and differences across all research areas.

In course of our study, we briefly describe each reviewed approach and compare them with a common schema, including features such as application domains, type of input data, number of input dimensions, applied TSA methods, aggregation or pre-processing of data instances, etc. We also analyze evaluations of the presented approaches to identify relevant data sets, which we also investigate and summarize. Based on the findings and limitations stated in the analyzed papers, we derive common problems and constraints that emerge when handling security data. We summarize our contributions as follows:

- A review of scientific publications leveraging time-series for cyber security applications,
- an overview of techniques from time-series analysis and how they are applied with security data, and
- a discussion of challenges, gaps, and recommendations for future research.

The remainder of the paper is structured as follows. Section 2 summarizes the background of time-series analysis for security applications. Section 3 provides a review of related literature. Section 4 outlines the methodology of our study, including our selection criteria for relevant publications. Section 5 provides an in-depth review of the selected publication, which we group by their application domains. Section 6 comprises a theoretical discussion of time-series models used in the literature and explains how they are applied with security data. Section 7 analyzes how authors evaluated their approaches and dives into publicly available data sets and evaluation metrics. Section 8 contains the answers to our research questions and recommendations for future work. Finally, Sect. 9 concludes the paper.

2 Background

This section covers the background of our research, including some general information on time-series analysis and the relevance of time-series in security analytics.

2.1 Fundamentals of time-series analysis

When analyzing time series—with the goal of predicting future values or gaining knowledge about the underlying process—mathematical modeling plays a crucial role. In general, two types of models can be distinguished: domain-specific models and general-purpose models. As a rule of thumb, domain-specific models are preferable when existing process information can be reflected in the mathematical model. For example, when modeling radioactive decay [93] or the spread of disease [15], a domain-specific model is a good starting point for data analysis. If this background knowledge about the domain or use case is not available, or should not be used, general-purpose models can be used for prediction. Many of these general-purpose models exist and are explained in standard textbooks [92].

A number of properties that can be differentiated along orthogonal dimensions are important for selecting the appropriate technique:

- **Area of influence.** A fundamental property of a model is the area of influence of the time series. A *global* model aims to describe and model the time series holistically; a *local* model, on the other hand, works with a segment, a so-called window, to model the data locally [47, 111].
- **Continuity.** The distinction between *discrete* and *continuous* can be approached in different ways [92, 109, 112]. In addition to mathematical definitions, applied mathematics adds a pragmatic perspective: in many cases, organizational, practical, or technical conditions determine whether a variable can be considered discrete or continuous, e.g., every computer measures time using a discrete integer, although time is considered continuous in most cases.
- **Model characteristics.** The model characteristics are determined by the starting point of the modeling [131, 144]. Two frequently used approaches with numerous hybrid forms can be distinguished: Model-based and data-based. In the *model-based* approach, the time series is modeled with a formula, a parametric model or a distribution. Due to its explicit character, it is possible to model prior knowledge. The purely *data-based* approach dispenses with an explicit model and describes the time series using similarities of itself. While the model-based approach ideally requires prior knowledge, sufficient historical data must be available in the data-driven case.
- **Fundamental patterns.** In the smooth transition of model properties, fundamental patterns assumed in the time series and to be considered in the model play an important role [92]. The most prominent representatives are cyclical and seasonal (a cycle occurs when the data show increases and decreases that do not have a fixed frequency, as opposed to seasonal patterns that always have a fixed and known frequency), as well as trend and stationary (a trend exists in a time series when there is a long-term increase or decrease in the data. The change need not be linear. A stationary time series does not depend on the time at which the series is observed. Consequently, time series with trends, seasonality, or cycles are not stationary).

2.2 The nature of time in security data analytics

The nature of time in time-series security data analytics is multifaceted and crucial for understanding and predicting various phenomena. Time-series data, captured at regular intervals, is fundamental in studying the dynamics of systems over time. This type of data is integral in a wide range of fields, including finance, healthcare, and security, among others. There are two important aspects when it comes to interpreting time in security analytics: use of time as feature in analysis and interpretation of time in analysis results.

2.2.1 Use of time as a feature in analysis

Leveraging time as a key feature provides crucial insights into patterns and anomalies, particularly evident in the field of time-series security analytics. Use of time as a feature can be seen from different perspectives [41]:

- **Incorporating Lag Features.** Lag features are a central aspect of time-series analysis. They involve using past values of a variable to predict future values. For example, in predicting a company's stock price, yesterday's closing price is a valuable piece of information. This approach captures temporal dependencies and trends, and is critical in fields like finance and economics. In security, previous user behaviour parameters, or network traffic statistics, could be a crucial information in determining anomalies.
- **Rolling Window Statistics.** This technique involves calculating summary statistics over a moving window of time. It's useful for smoothing out noise and focusing on underlying trends. The window size and the specific statistics used (like mean or standard deviation) depend on the application and the characteristics of the data. Rolling window features are valuable for identifying long-term trends, seasonal patterns, and sudden shifts in data.
- **Time-based Features.** Features like the day of the week, the month of the year, and holiday indicators are critical for capturing seasonality and temporal patterns in data. For instance, in security analytics, data volumes exchanged over network may vary significantly between weekdays and weekends or during specific holiday periods. Extracting these time-based features can greatly enhance the accuracy of predictive models in time-series analysis.
- **Cyclic Time Features.** Previously mentioned time-based feature (day in month or in week, the month of year...) does not, from the analytic model point of view, formally reflect cyclic nature of time, and the fact that e.g. value that comes after $Sunday$ is $Monday$. Additional pre-processing and feature engineering techniques are usually necessary to address these issues, for example, by applying sine and cosine transformations on the total number of seconds passed since a reference time point that effectively presents time in 2-dimensional space similar to a clock (see Stojanović et al. [113] for a detailed description and equations).

2.2.2 Interpretation of time in results

The interpretation of time in time-series analysis results is primarily about understanding the dynamics and underlying patterns of the data. It involves:

- **Identifying Trends and Seasonality.** By analyzing time-series data, one can identify trends (long-term increase or decrease) and seasonality (regular, predictable patterns within a specific time frame). This understanding is crucial in fields like security, but also meteorology [76] and economics [87], where recognizing patterns over time can inform future predictions and strategies [28].
- **Understanding Temporal Dependencies.** Time-series analysis allows for the understanding of how current values are influenced by past values. This temporal dependency is key to predicting future events based on historical data [92].
- **Interpreting Feature-Based Representations.** In feature-based time-series analysis, various representations are used to interpret the data meaningfully [42]. Features might encode deeper theoretical concepts like entropy or stationarity, or they might derive from the shapes of time-series subsequences. Understanding these features can provide insights into the characteristics of different classes within the data, thus aiding in more accurate predictions and better domain understanding.

3 Related work

Anomaly detection for cyber security applications has been widely researched in the past. Accordingly, several comprehensive surveys already exist. Pang et al. [94] and Chalapathy et al. [22] review deep learning approaches for anomaly detection and specifically focus on the various types of neural network models suitable for certain application cases, including cyber security. Fernandes et al. [37] survey approaches for anomaly detection in network traffic data and thereby identify several classes of data, anomalies, and detection techniques. Even though most of these surveys mention time-series as a way to represent data and modeling techniques for anomaly detection, they do not delve into the various application domains of cyber security and instead pursue a broad overview of the topic of anomaly detection. Accordingly, these surveys are unable to identify similarities and differences among these application areas and cannot analyze and compare domain-specific aspects such as data properties, suitable models, and commonly encountered issues with respect to certain use-cases.

Forecasting, on the other hand, is a research topic that is often realized with time-series models; however, existing surveys on forecasting, such as the one by Torres et al. [120] that focuses on deep learning techniques, do not consider any application contexts such as security. To the best of our knowledge, our paper is the first survey that particularly focuses on time-series analysis for both anomaly detection and forecasting in the cyber security domain.

Given the deep scientific background of time-series analysis and the widespread utilization of methods thereof, it comes at no surprise that researchers have studied the application of time-series analysis in various domains, including cyber security. For example, Choi et al. [24] provide a comprehensive survey of anomaly detection methods using deep learning models suitable for time-series data. Their application ranges cover a broad range of topics that are relevant for industrial settings, such as detection of damage or faults in smart manufacturing processes, resource optimization through modeling of energy consumption patterns, and monitoring of cloud computing environments. While the authors discuss intrusion detection as one of the application scenarios, they only provide a brief discussion of few related publications but do not provide any details on the various types of data encountered in security applications. Moreover, they do not consider time-series models other than those based on deep learning. Our literature review, on the other hand, considers all types of time-series models and provides a more comprehensive view on cyber security topics that also go beyond intrusion detection.

The survey by Braei et al. [18] considers statistical models, machine learning, and deep learning for anomaly detection. They compare the analyzed methods on various data sets such as network traffic captures. Even though this kind of data is relevant in security applications, the authors do not consider the various types of attacks that could manifest in network traffic and carry out their analysis without considering the application context. Moreover, the survey only includes univariate time-series methods, which limits the scope to approaches and data sets that fulfill this requirement. In opposite to that, our paper emphasizes the diversity of application contexts and their respective requirements and considers both univariate as well as multivariate methods.

Internet-of-Things (IoT) devices are often deployed to monitor physical attributes and thereby produce time-series data suitable for analysis. Since anomaly detection enables the automatic identification of issues, many researchers focus on time-series analysis on IoT data. Sgueglia et al. [105] conduct a systematic literature review and identify several open research gaps, including problems with high-dimensional data and lack of robust and generally applicable models suitable for practical use-cases. Cook et al. [27] provide another survey on this topic and state that efficient, incremental, and adaptive models are required to enable utilization of time-series analysis in real-world applications. Rather than focusing on IoT explicitly, we identified industrial control systems as a relevant application area, among several others. We therefore discuss the related topics IoT and cyber-physical systems as part of that application area. Moreover, both surveys touch the subject of cyber attacks as a possible cause of anomalies, but are generally directed towards generic anomaly detection. Our survey specifically analyzes requirements in the context of cyber security.

Other than aforementioned surveys that only marginally focus on aspects of cyber security, Husák et al. [51] provide a survey on attack prediction, which includes attack projection (predict the next move of an attacker), intrusion prediction (predict upcoming cyber attacks), and forecasting of the cyber situation as a whole. Thereby, they provide in-depth explanations of various models commonly used in literature and state shortcomings, such as the need for attack models and the lack of openly available data sets suitable to evaluate the proposed approaches. Other than our paper, the survey aims to provide an overview of attack prediction approaches using any type of modeling or learning technique, thus only some of the reviewed approaches leverage methods from time-series analysis.

In addition to surveys or review papers, some publications explain and compare several different methods from time-series analysis on the same data sets in order to investigate the capabilities of these models for a certain kind of data. For example, Lande et al. [68] evaluate four forecasting models on four different data sets, Kalouptsoglou et al. [58] compare the prediction capabilities of models based on deep learning and statistical models, and Pokhrel et al. [96] compare linear with non-linear models. In the following, we review the approaches analyzed in these papers based on their application domains; in Sect. 6, we individually describe each of the time-series models encountered in the reviewed literature.

4 Methodology

To answer the research questions stated in Sect. 1, we conduct a review of scientific publications that apply methods from time-series analysis in the context of cyber security applications. To this end we gather a set of publications using a web search on Google Scholar[1] using the keywords "cyber security", "intrusion detection", "cyber attacks", "IDS alerts", and "cyber situational awareness" in combination with "time-series" and "forecasting". We point out that it is infeasible to survey the intersection area of cyber security and time-series analysis completely, since any publication that analyzes timestamped security data can be argued to be related to time-series analysis. To overcome this problem, we pursue a broad review of relevant publications and therefore only include scientific papers that explicitly focus on the application of methods from time-series analysis using temporal security data. Note that this excludes (i) papers that only visualize time-series for interpretation without applying any technical methods, such as trend analyses for malware [1] or vulnerabilities [23, 30], (ii) papers that do not make use of time-dependent data even when methods from time-series analysis are applied, such as shapelet detection in executable files [95], (iii) papers that do not process security data, such as detection of video manipulations [140], as well as (iv) papers on adversarial attacks [36, 59, 98].

We analyze each of the selected publications with respect to a set of pre-defined features that are suitable to distinguish different approaches in time-series analysis. In the following, we briefly describe each of these features.

- **Domain** states the corresponding application domain for the approach presented in the respective publication.
- **Variables** states whether multiple features are used by the time-series models, i.e., whether univariate (UV) analysis that only considers a single variable or multivariate (MV) analysis that considers more than one variable is carried out. Some approaches consider multiple features but only analyze them individually with univariate methods, which we mark with an asterisk (UV*).
- **Data** specifies the type of data used in the publication.
- **Features** describes the input derived from the data that is fed into the time-series models.
- **Values** specifies whether the input data involves continuous (Con), discrete (Dis), categorical (Cat), or binary (Bin) features.
- **Method** states the time-series models or analysis methods used in the publication.
- **Granularity** states whether analysis is based on aggregated data such as event windows (Window) or time-windows (TW), single observations (Point), or time intervals (Interval). For event windows and time windows we also state the length of the window as the number of events or temporal duration respectively. This distinction is based on the classification presented by Li et al. [72].
- **Seasonal** specifies whether the publication considers seasonality as part of the analysis, and the lengths of the periods used in the approaches.

We realize that a large portion of our gathered publications specifically focuses on the detection of anomalies in security data. This is reasonable, since the automatic detection of immanent threats is a widely researched topic in cyber security and time-series analysis enables modeling of system behavior patterns and the recognition of deviations from these models. For these publications, we therefore additionally assess the following features.

[1] https://scholar.google.com/.

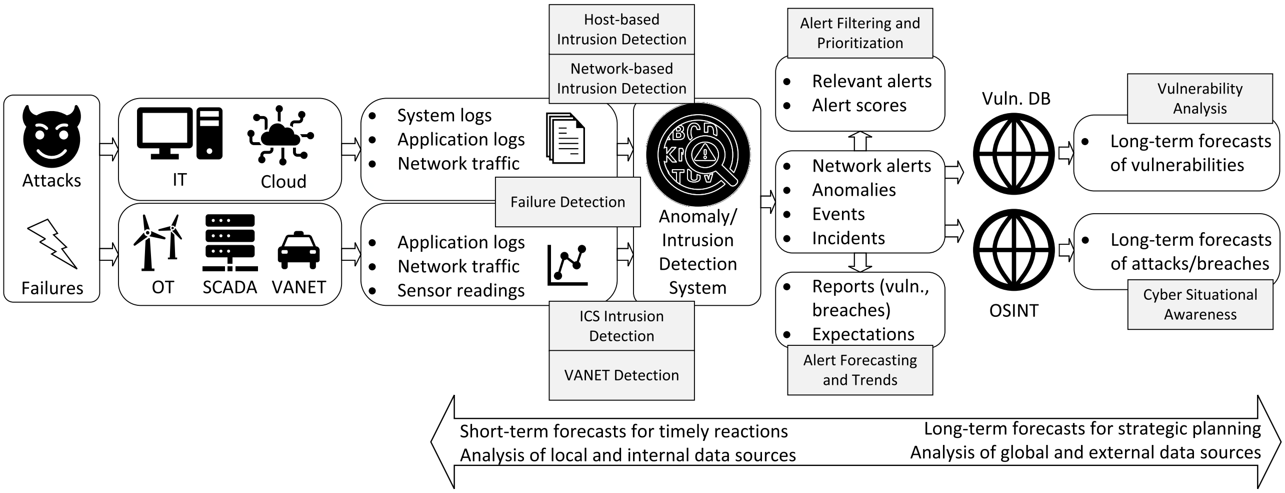

Attacks
Failures
IT
Cloud
OT
SCADA
VANET
System logs
Application logs
Network traffic
Application logs
Network traffic
Sensor readings
Host-based Intrusion Detection
Network-based Intrusion Detection
Failure Detection
ICS Intrusion Detection
VANET Detection
Anomaly/ Intrusion Detection System
Alert Filtering and Prioritization
Relevant alerts
Alert scores
Network alerts
Anomalies
Events
Incidents
Reports (vuln., breaches)
Expectations
Alert Forecasting and Trends
Vuln. DB
OSINT
Vulnerability Analysis
Long-term forecasts of vulnerabilities
Long-term forecasts of attacks/breaches
Cyber Situational Awareness
Short-term forecasts for timely reactions
Analysis of local and internal data sources
Long-term forecasts for strategic planning
Analysis of global and external data sources

Fig. 1 Overview of common application domains (grey boxes) leveraging time-series analysis along various stages of cyber security analytics. Attacks and failures in IT or OT systems are disclosed in technical log data by fast, autonomous, and effective anomaly detection techniques. Subsequently, generated alerts are either filtered and prioritized for further manual analysis, or form the basis for long-term forecasts of cyber threats

- **Anomalies** explains the type of anomalies in the data. For example, anomalies could be related to brute force intrusions, denial of service attacks, manipulation of data, or system failures without malicious intent.
- **Mode** specifies whether training the anomaly detection system is supervised (SUP), i.e., labels for normal and anomalous instances are required, semi-supervised (SEMI), i.e., only normal instances are used for training, or unsupervised (UNS), i.e., no label information is required.

The following section provides an overview of the reviewed publications, which we group by their application domains. The section also contains tables that state each of the aforementioned features for every publication.

5 Application domains

Cyber security specialists leverage methods from time-series analysis in several highly diverse application domains, each with their own goals, requirements, and data types. Based on the reviewed papers, we identify some of the most prevalent application areas and investigate the use of TSA separately for each domain. We first provide an overview and then describe some approaches of each domain in detail.

5.1 Overview of time-series analysis in cyber security analytics

Time-series analysis comprises generic methods that are applicable in many different fields of cyber security analytics. In course of this literature review, we identified the main application areas where time-series analysis is applied. Figure 1 provides an overview of the workflow of cyber security analytics, where each of our identified application domains is placed as a grey box alongside a sequential chain of cyber security analytics that spans from local and timely threat detection in high-frequency log data to global and long-term analysis of attacker behavior patterns.

On the left hand side of the figure, security incidents caused by malicious actors that aim to compromise networks, exfiltrate data, or damage systems, as well as safety events caused by failures or other system problems, affect both IT networks prevalent in enterprises (servers, hosts, etc.) as well as cyber-physical systems. Monitoring solutions that maintain a record of a wide variety of events in the form of system logs, network traffic captures, application log data, sensor readings, etc., contain traces of these undesired activities and are thus suitable data sources for detection of any adverse activities. Methods from TSA are suitable candidates for this kind of anomaly detection, specifically when it comes to modeling periodically occurring events and disclosing unusual changes of behavior patterns such as sudden increases of event frequencies. We differentiate between methods leveraged by *host-based intrusion detection systems (HIDS)* that analyze system and application log data collected from servers and hosts, *network-based intrusion detection systems (NIDS)* that analyze network traffic and packet captures in organizational networks, *ICS intrusion detection (ICS)* that analyzes network traffic and sensor readings from cyber-physical systems, *VANET detection (VANET)* that leverages data from vehicular ad-hoc networks, and *failure detection (FD)* that aims to disclose any unusual deviations that may relate to system issues.

The amount of alerts that cyber security analysts need to deal with often becomes overwhelming, mainly due to the facts that attacks trigger multiple alerts at once and at the same time false positive alerts resulting from unusual but otherwise benign normal behavior are frequent [5]. To counteract this issue, analysts leverage TSA to identify alerts that stand out from regularly occurring alert patterns as they are likely more relevant and thus require higher attention by operators [128]. The application domain *alert filtering and prioritization (AFP)* deals with the identification and scoring of such relevant alerts.

On the other hand, analysts may be interested in forecasting the numbers and patterns of alerts that can be expected to occur in the near future based on the current stream of alerts observed in their local infrastructure. This is accomplished by *alert forecasting and trends (AFT)*, which focuses on short-term forecasts and the detection of attacks that cause bursts of alerts, e.g., brute force attacks that trigger a high number of alerts in a short period of time for every failed login attempt.

Successful attacks that are relevant to a broader community are eventually distributed through threat intelligence platforms, which provide another source of information that is suitable for time-series analysis. We identify the domain of *vulnerability analysis (VA)*, which crawls vulnerability databases storing the dates when vulnerabilities were found for certain software versions. Analysts apply TSA to predict the number of expected vulnerabilities up to several months into the future [135]. On the other hand, publicly available information on the dates and scales of large-scale attacks is used to predict the frequencies and severities of attacks in the future. To this end, the *cyber situational awareness (CSA)* research domain leverages threat intelligence such as attack reports as well as open-source feeds such as news items to estimate the probability and severity of upcoming cyber attacks.

In the following, approaches that have been proposed by the scientific community to address each of these application domains are reviewed and summarized. Tables 1, 2 and 3 provide an overview of all reviewed approaches focusing on anomaly detection as well as modeling and prediction of time-series respectively, where the columns in the tables correspond to the features described in Sect. 4.

5.2 Failure detection

If configured correctly, almost all actively used IT components produce log data or network traffic in one or another way. A main motivation to create and store these permanent records of system activities is to provide operators a verbose source of information during problem investigations where logs are analyzed in hindsight, in particular, when it becomes necessary to identify root causes by tracing back at what point in time and under which circumstances certain erroneous states occurred [31, 66]. Anomaly detection also aims to disclose any undesired states, but is generally applied in an online manner where logs are analyzed continuously almost at the same time as they occur, thereby enabling timely detection of system problems and alleviating the need for manual and purely forensic analysis. Approaches relying on machine learning techniques have proven itself as a highly useful concept for anomaly detection as it is able to efficiently process large amounts of data and derive complex behavior patterns that enable the detection of rare or unusual activities that stand out from the log events corresponding to normal system behavior [81].

Failure detection is often closely related to cyber security. In particular, security-relevant incidents may produce artifacts that are also encompassed by common failures and only later on recognized as the result of malicious activities [66], such as unexpectedly crashing processes, unusual system behavior after accidental misconfiguration of services, over-utilization of system resources, etc. Failure detection can thus be regarded as the first line of defence when it comes to technical detection of intrusions, before more advanced and specialized solutions come into play.

5.2.1 From logs to time-series

Log events generally contain a time stamp that indicates their time of generation and allows to order them chronologically [65]. As such, log data effectively forms time-series, even though data points are usually not spaced equally in time. In addition to the time stamp, log lines usually comprise a set of continuous, categorical, and discrete parameters extracted from the events as well as information on the type of event itself. Note that determining the event type as well as extraction of parameter values generally requires log parsing, i.e., matching of each log line to a set of templates that specify the syntax of log messages and provide a mapping for parameter values to variables [66, 139].

One of the most widely used methods to transform logs into an evenly spaced time-series with purely continuous values is to generate event count vectors. Thereby, a time window of fixed size is moved in regular step sizes over all data points and distinct events within that window are counted [46, 61, 67, 128, 141]. Ohana et al. [89] apply this approach on parsed cloud logs and leverage the resulting event count vectors for unsupervised anomaly detection. In particular, they apply Exponentially Weighted Moving Average (EWMA) and standard deviation to predict the mean and expected deviation of expected counts for log events from historic data. In addition, they also compute the deviation of predictions based on two different time window lengths in order to recognize when the prediction capability of their model deteriorates. They also present an anomaly score that is based on the deviation of expected and actually observed

Table 1 Overview of anomaly-based approaches for low-level event data

Approach	Dom.	Var.	Data	Features	Val.	Anomalies	Method	Gran.	Mode	Seas.
[31]	FD	UV*	Application logs	Event counts	Con	Failures	Clustering	Window (20 elem.)	UNS	No
[56]	FD	MV	Application logs	Event counts	Con	Failures	Clustering	TW (5 min.)	UNS	No
[73]	FD	MV	Application logs	Event sequences, inter-arrival times	Con	Failures	RNN (LSTM)	Window (20 elem.)	SUP	No
[81]	FD	MV	Application logs	Event sequences, event counts	Con	Failures	RNN (LSTM)	TW (20 elem.)	SEMI	No
[89]	FD	UV*	IBM Cloud Syslog	Event counts	Con	Failures	EWMA, Gauss. Tail	TW (5 min.)	UNS, SUP	No
[143]	FD	MV	Application logs	Event counts, seasonality	Con	Failures	RNN (LSTM)	Window (1–10 elem.)	SUP	Yes
[9]	HIDS	MV	Web logs	Error rates	Con	Generic attacks	βARMA	TW (10 min.)	SEMI	No
[21]	HIDS	UV	Syscall logs	Event sequences	Cat	Event traces of malware	CNN, RNN (LSTM, GRU)	Time-series	SUP	No
[44]	HIDS	MV	Web logs	Event counts (status codes)	Con	DoS, brute force (simulation)	Clustering	TW (10 s, 60 s)	UNS	No
[60]	HIDS	MV	Key access logs	Event counts	Con	Generic anomalies	GAN, LSTM	TW (5–30 min.)	SEMI	Yes
[64]	HIDS	MV	HW perf. counters	Mean, slope, standard deviation	Con	Malware	Conventional ML	TW	SUP	No
[67]	HIDS	UV*	MySQL logs	Event counts	Con	Changed event frequencies, peaks, missing events	ARIMA	TW (15 min.)	UNS	No
[86]	HIDS	UV	Network monitoring data	Generic	Con, Dis	Generic anomalies	Markov chains, DLM	Point	UNS	Yes
[107]	HIDS	UV	Web logs	SOAP Message size	Con	Oversized payloads, DoS attacks	ARIMA	Point	SEMI	No
[119]	HIDS	UV	Web servers, user machines	Event counts, CPU and memory usage	Con	Generic anomalies	SVM, LSTM, MA, EWMA, Percentile	TW, point	UNS, SEMI	No

Table 1 continued

Approach	Dom.	Var.	Data	Features	Val.	Anomalies	Method	Gran.	Mode	Seas.
[130]	HIDS	UV	Malware executables	Outbreak time-of-day	Con	Malware signatures	RNN (LSTM)	Interval	SUP	No
[133]	HIDS	MV	Linux server logs	Event counts	Con	Generic attacks	RNN (GRU)	TW (60 min.)	UNS	Yes
[7]	ICS/ HIDS	UV, MV	Transactions, water treatment, application logs	Amounts, status codes, etc.	Con	Various (fraud, tampering, etc.)	LOF, DT, PCA, RNN (LSTM-AE)	TW (5–60 sec.), point	UNS	No
[104]	ICS/ HIDS	MV	PCAPs, CPS systems	Netflow data, sensor readings	Con	DoS, exploits, worms, etc.	RNN (GRU)	Point	SEMI	No
[79]	NIDS	MV	Network traffic data	Netflow data	Con, Dis, Bin	DoS, U2R, R2L, probing	RNN (LSTM)	Point	SEMI	No
[74]	NIDS	MV	Network traffic data	Netflow data	Con, Dis, Bin	DoS, U2R, R2L, probing	CPA-TCN	Point	SEMI	No
[2]	NIDS	MV	Network traffic data	Netflow data	Con, Dis, Bin	DoS, DDoS, recon., information theft	DNN	Point	SUP	No
[38]	ICS	MV	Gasoil heating model	Sensor readings (tank level, temperature)	Con, Bin	Unauthorized changes (maximum temp., frequencies, etc.)	RNN (LSTM)	Point	SEMI	No
[49]	ICS	MV	Aviation testbed, robot	Codes (message type, addresses), sensor readings (speed, voltage)	Cat, Con	DoS, noise, protocol violation, buffer, payload, sensor obstruction	HMM	Point	UNS	No
[80]	ICS	MV	Water distribution plant	Sensor readings, water levels, pumps	Con, Bin	Signal manipulation, replay attacks, concealment	DT, kNN, SVM	TW (6–48 hrs.)	SUP, UNS	No
[70, 71]	ICS	MV	Water treatment plant	Sensor readings, actuator status codes	Con	Tampering with compromised sensors	RNN (LSTM), GAN	TW (2 min.)	SEMI	No
[137]	ICS	MV	Water treatment plant, gas pipeline	Sensor readings	Con, Dis	Tampering with compromised sensors	GAN	Point	UNS	No
[115]	VANET	MV	CAN data frame	Destination identifier, data value (e.g., speed), time frame	Con, Dis	Data manipulation, flooding	RNN	Point	SUP	No
[136]	VANET	MV	Traffic sim	Vehicle speed, acceleration, etc.	Con	False messages, collusion	RNN (LSTM)	Interval	SUP	No

Table 2 Overview of anomaly-based approaches for alert data and threat intelligence

Approach	Dom.	Var.	Data	Features	Val.	Anomalies	Method	Gran.	Mode	Seas.
[127]	AFP	UV*	Alerts from IDSs	Alert counts	Con	Generic anomalies	EWMA	TW (1 h)	UNS	No
[129]	AFP	UV*	Alerts from IDSs	Alert counts	Con	Generic anomalies	AR	TW (1 h)	UNS	Yes
[128]	AFP	UV*	Alerts from IDSs	Alert counts	Con	Generic anomalies	NAR	TW (1–20 min.)	UNS	Yes
[142]	AFP	MV	Alert management system	Event count, frequency, seasonality, inter-arrival time	Con	Severe alerts	RNN (LSTM)	TW (15–30 min.)	SUP	Yes
[61]	AFT	UV	Threat Intelligence platform	Alert counts	Con	Changes of mean or trend	ARIMA	TW (1 day)	UNS	No
[108]	AFT	UV	Alerts from IDSs	Event counts	Con	Brute force, DoS	EWMA	TW (1–6 hrs.)	UNS	No
[33]	CSA	MV	Breach reports	Loss severity and frequency	Con	Change points of overall trend	GLM, GDM, GLR, tests	TW (1 month)	UNS	Yes
[90]	CSA	UV*	Online sentiment signals, Twitter, GDELT	Event counts, sentiment scores	Con	Generic attacks	ARIMA	TW (1 day)	SUP	No

Table 3 Overview of analytical approaches for alert data and threat intelligence

Approach	Dom.	Var.	Data	Features	Val.	Method	Granularity	Seasonal
[13]	AFT	UV	Malware reports	Event counts	Con	BSSM, AR	TW (1 week)	No
[50]	AFT	UV	Alerts from sharing platform	Event counts	Con	ARIMA, ETS	TW (5–60 min.)	Yes
[8]	AFT	MV	Alerts from sharing platform	Time, Flows, Ports, IPs, Protocols, Categories	Con, Dis	RNN (GRU)	Window (12 elem.)	No
[32]	CSA	UV*	Data breach reports	Breach sizes and breach inter-arrival times	Con	QAR, CQAR, ARMA-GARCH	Point	No
[68]	CSA	UV	Event mentions in news, blogs, etc.	Event counts	Con	LSTM, ARIMA, LPPL, N-gram	TW (1 day)	No
[100]	CSA	UV	Cyber attack reports (Imperva)	Event counts	Con	RNN, CNN, ARIMA, XGB, RF, SVM, KNN	TW (5–30 min.)	No
[132]	CSA	UV*	Public attack report database	Attack counts	Con	ARIMA	TW (1 day)	No
[138]	CSA	MV	Financial losses of companies	Date, loss, company, region, event type, etc.	Con	LR, CPD	TW (1 year)	No
[141]	CSA	MV	Security events	Event counts	Con	RNN (LSTM, GRU, mLSTM), EVT	TW (1 h, 1 day)	No
[58]	VA	UV	Vulnerability reports for software and OS	Vulnerability counts	Con	SES, TES, ARIMA, Croston, MLP, RNN (LSTM, BiLSTM, GRU), CNN	TW (1 month)	Yes
[96]	VA	UV	Vulnerability reports for OS	Vulnerability counts	Con	ARIMA, FFNN, SVM	TW (1 month)	No
[135]	VA	UV	Vulnerability reports for software and OS	Vulnerability counts	Con	SES, DES, TES, ARIMA, Croston, FFNN	TW (1 month)	Yes

event counts as well as the model prediction capability. While this approach is fully unsupervised, they also discuss a supervised prioritization scheme that is based on the occurrence time and location of the event, which is matched against events from historic tickets with known relevance.

5.2.2 Clustering and outlier detection

Jain et al. [56] point out that a single failure in the network may trigger logs in multiple different components, which are difficult to relate as the event types and contents of logs from diverse applications usually do not coincide. Other than Ohana et al. [89] who conduct prediction separately for each host, they create event counts across all components and apply Agglomerative Hierarchical Clustering (AHC) with the Jaccard metric as a similarity function to identify correlating event types from the individual time-series. Another approach that clusters event count vectors is presented by Du et al. [31]. Even though they also consider multiple different event types, each of them is analyzed separately and thus correlations between event types are neglected. They analyze the time-series of event count occurrences for spikes, i.e., sudden increases of log frequencies in single time windows in comparison to nearby windows. To this end, they apply a sliding window approach on the time-series of count vectors for each event and cluster the resulting sub-sequences using Euclidean distance. They also compute an anomaly score for log segments comprising several event types, which bases on the dissimilarity of sub-sequences of all involved event types as well as the anomaly score of the most similar sub-sequence from the training data.

5.2.3 Neural networks

There also exists a significant number of publications that leverage neural networks for anomaly detection in log data; we refer to the survey by Landauer et al. for a detailed investigation of this research area [65]. In addition to count vectors, neural networks are also capable of ingesting sequences of categorical log event occurrences directly; Recurrent Neural Networks (RNN) such as Long Short-Term Memory (LSTM) RNNs are designed to learn sequential patterns, which makes this kind of neural network architecture especially suited for chronologically ordered data with strong sequential dependencies such as logs. For example, Meng et al. [81] use a combination of count vectors and sequential features for detection of anomalies with LSTM RNNs. In particular, given a sequence of events, the neural network predicts the probabilities of subsequently following event and raises an anomaly if the actually observed event is not among the expected ones.

While the time stamp is most often only used to ensure chronologically correct ordering of events but not fed into neural networks, some approaches also leverage the inter-arrival time of events for the purpose of detecting temporal changes of event occurrences, such as delays. For example, Li et al. [73] generate a time-series for event inter-arrival times and feed the embedded time-series together with transformed event sequences into an LSTM RNN. Moreover, Zhou et al. [143] point out that log events often occur periodically, causing that count vectors form seasonal time-series. They derive some statistics on seasonal logs and use them together with event counts as features for an LSTM RNN.

5.3 Host-based intrusion detection

In the previous section, anomalies correspond to traces of system failures in log data that stand out from traces occurring during normal behavior in terms of event occurrences, frequencies, sequence orders, etc. Generally speaking, most anomaly detection techniques applied for the purpose of intrusion detection monitor and analyze the same features as approaches for failure detection do, because attack manifestations are assumed to have similar effects on log data. Accordingly, generation of event count vectors is still the preferred way to convert categorical event logs into time-series. However, while approaches for failure detection primarily leverage application logs that contain failures of the respective applications, intrusion detection approaches leverage data sources that are known to be affected by certain types of attacks. In the following, we go through some of the most common data sources that have been used for this purpose in existing works. Note that data sets other than log data that are commonly used by host-based intrusion detection systems, such as searches for hashes in file systems, databases, or registries, as well as forensic memory analyses, usually do not involve time-series and are thus not included in our review.

5.3.1 Access logs

Access logs are one of the most common data sources when it comes to the detection of attacks on IT infrastructure. The logs usually document which user requests which resource at what point in time, and whether this attempt was successful or not. The latter information is usually provided via categorical status codes, which are a focus point of several publications. Granlund et al. [44] create count vectors of these status codes analogous to event count vectors as discussed in the previous section. They apply several clustering algorithms on the count vectors, but treat them as independent instances so that their times of occurrence or temporal dependencies are neglected for detection. Nonetheless, the approach was able to detect time windows where some status codes were changed or their frequencies increased as part of a manual injection procedure.

Anastasiadis et al. [7] present another approach that analyzes status codes, among other data sets on topics such as fraud detection and server monitoring. They apply various anomaly detection techniques, including methods specifically designed for outlier detection such as Local Outlier Factor (LOF) and Isolation Forests (IF), but also neural networks. Other than status codes, web logs also contain the size of transmitted objects, which is relevant for cyber security as oversized payloads are a common method to achieve Denial of Service (DoS), brute-force, spoofing, or flooding in web applications. Shirani et al. [107] therefore use an ARIMA model to forecast the payload size in the Simple Object Access Protocol (SOAP) and detect outliers if the actually observed size is outside of the expected confidence interval.

Other approaches consider the number of access logs rather than specific parameters within them. Khoshnevisan et al. [60] count the number of access logs in multiple time windows to generate a multivariate time-series, which they feed into a Generative Adversarial Network (GAN). They acknowledge that real log data is usually highly seasonal due to the cyclic nature of human work time and thus integrate seasonality as an attention mechanism for their neural network. Another approach based on the total number of generated web logs is presented by Ara et al. [9], who make use of two error rates measured in time windows of 10 minutes. Their bivariate βARMA model allows to predict the rates up to 6 time windows or 60 minutes ahead and provides confidence bounds for anomaly detection. While they focus on the special case of two variables, they point out that their methodology is suitable for multivariate time-series with arbitrary many variables.

5.3.2 Application logs

Application logs are less structured than access logs; accordingly, they either require parsing before event count vectors can be generated. The approach by Landauer et al. [67] avoid the need for parsers by clustering the logs within time windows and observing the evolutions of clusters over time; thereby, cluster sizes yield time-series that are predicted one step ahead after every time window to identify anomalies that are outside of the prediction interval. While multiple evolving clusters exist at the same time, they analyze each time-series individually in an univariate fashion, but suggest to analyze correlation between time-series for future work. Wu et al. [133] create a multivariate time-series by counting events from five Linux system processes (e.g., crond or sshd) and feed them into a RNN. An interesting variation of their approach is that they append seasonal decompositions of the time-series to the input of the neural network, emphasizing that log data is often affected by multiple seasonal influences at the same time.

5.3.3 System calls and low-level system data

Similar to approaches from failure detection such as the one by Meng et al. [81], analysis of system calls is often carried out with focus on sequential patterns in log data. Čeponis et al. [21] claim to leverage univariate time-series but omit the time stamps when feeding chronologically ordered sequences of system calls into three distinct neural network architectures LSTM RNN, GRU RNN, and CNN to differentiate malware samples from benign ones. Kuruvila et al. [64] detect malware through hardware performance counters that monitor low-level events such as CPU-cycles and cache-misses. The authors demonstrate that temporal attributes, which are often neglected when analyzing this kind of data, are effective for malware detection and suitable to be analyzed as sequential time-series. Specifically, they leverage time-series forest, which splits time-series into intervals, computes statistical data such as the mean, slope, standard deviation, etc., and trains machine learning models on the chunks. Another source of data are resource monitoring logs as shown by Tiwari et al. [119]. They leverage numeric metrics such as CPU and memory usage that are suitable to detect situations where system resources are acquired by attackers, which is a particularly relevant attack scenario in cloud computing environments.

Wang et al. [130] also focus on malware samples classified with LSTM RNNs; however, the application of time-series comes from the fact that outbreak times are included in their training data, which allows them to predict the times of day where malware occurrences are expected. The main reasoning behind this is that more resources can be allocated to scanning activities during these hours in comparison to time intervals that are considered safe. For an in-depth survey on time-series analysis for malware classification, we refer to the study by Finder et al. [39], whose framework incorporates conventional machine learning methods for temporal pattern mining as well as neural networks that are used to process categorical events.

5.3.4 Generic log data

Naveiro et al. [86] present a generic approach that is applicable to many features found in network monitoring logs, such as resource utilization metrics. They cover continuous features for which they use combinations of linear and seasonal terms, and discrete features for which they suggest Markov chains. On top, they have a separate model that is able to predict outbursts, i.e., rapid increases of event counts. Their main motivation for selecting these techniques for time-series analysis is that commonly applied methods such as ARIMA, neural networks, or statistical models do not adequately support scalable and automated analysis as manual adjustments are almost always required. Another unusual attribute of their approach is that it is designed to produce both a short-term forecast for the detection of point outliers as well as long-term forecasts to estimate values in future time intervals.

Finally, we also mention two commercial products for log data analysis that leverage time-series analysis for anomaly detection. IBM's QRadar[2] involves so-called behavioral rules that monitor event frequencies over a manually defined period of time (e.g., 1 week) and then predict ranges in which event frequencies are expected, reporting alerts when newly observed frequencies are too far outside of the intervals, which is steered with another manually defined threshold parameter. Computing the forecast is simple and only depends on a base value, the current trend of the time-series, and the seasonal adjustment. DataDog[3] provides a basic model without seasonality, a robust seasonal-trend decomposition model that adjusts for seasonality but takes longer to adjust to long-term shifts, and an agile model that relies on SARIMA to quickly respond to level shifts.

5.4 Network-based intrusion detection

The next most common approach in time-series analysis in security analytics is network-based intrusion detection. This type of intrusion detection approach typically analyses network traffic and packet captures in enterprise network ICT environments. The original raw network data consists of raw IP packets with various headers based on higher network stack protocols. Due to its size and variability, the use of this raw format in automated IDS approaches is impractical. Consequently, most published methods use time based feature construction as a critical pre-processing step.

Constructed features not only speed up detection, but also improve the discriminative power compared to the original raw network data. This significant improvement benefits machine learning algorithms, making their use feasible. Feature construction methods include manual approaches or data mining techniques such as sequence analysis, association mining and frequent episode mining [11]. Network based features are usually categorised by type: binary (e.g. flag state), numeric (e.g. number of bytes) and nominal (e.g. protocol type). Regardless of the final choice of features, the common characteristic is that these features are always sampled in time and contain time stamps, making them time series.

Intrusion detection systems can be divided into two main categories: classification-based and anomaly-based. Classification-based IDSs use machine learning algorithms to categorise incoming data based on defined characteristics. While they are effective against known attacks, they may struggle with new, unknown attacks that have no correlation to the training dataset. Conversely, anomaly-based approaches use statistical models to establish a baseline of normal behaviour and identify deviations. This allows them to detect unknown attacks, but may yield lower results when encountering packets from a previously unseen class [94].

When it comes to IDS approaches addressing specifically time-series, most publications aim at Time Series Classification (TSC). TSC is a subset of machine learning, characterised by its requirement for ordered data samples, in contrast to traditional classification tasks [77].

When addressing intrusion detection as a time series challenge, many Deep Learning (DL) approaches focus on recurrent network architectures such as Gated Recurring Units (GRUs) and Long Short-Term Memory (LSTM) [12, 74]. These networks inherently capture temporal information from data. For example, Mahdavisharif et al. [79] introduced a model that uses the LSTM architecture to detect complex intrusion patterns in a massive amount of traffic.

Although methods using recurrent network architecture can achieve good results, they are notable for their complexity and have certain limitations in detection performance, as indicated by Li et al. [74]. Ahmad et al. [2] conducted a comparative study of DL methods, including RNNs, LSTMs, GRUs, and CNNs, highlighting the complexity of recurrent architectures.

5.5 ICS intrusion detection

Other than host-based intrusion detection where analyzed data often consists of categorical event types that require counting and network-based intrusion detection where approaches rely on feature construction, ICS intrusion detection generally leverages data sources recording continuous features such as sensor readings that are suitable to be directly used for the purpose of model building and detection. However, categorical features also appear in the form of actuator states and need to be handled by the algorithms. Data sets are often collected in cyber-physical test environments or through simulations, which represent industrial processes such as water treatment plants or gas pipelines. Since these processes involve a high number of sensors and actuators that are all part of the same control loop, the data sets usually involve many features with strong dependence between them. Accordingly, approaches are usually designed for multivariate analysis of high-dimensional data.

5.5.1 Continuous sensor readings

A common use case in ICS intrusion detection is that of Secure Water Treatment (SWaT) [43], which is a well-known data set collected in a simulation testbed. The test environment comprises data sources such as water level sensors and involves attacks where sensors are manipulated with the aim of over- or underflowing tanks and damaging pumps. For

[2] https://www.ibm.com/qradar.

[3] https://docs.datadoghq.com/.

example, Li et al. [70] propose an unsupervised approach based on GANs and LSTM RNNs that analyzes the SWaT data set and provides anomaly scores for time windows of 2 min. One of their main insights is that a multivariate anomaly detection approach is clearly able to outperform an univariate approach in terms of false alarm rates, precision, and recall. The authors also apply a similar approach on Water Distribution (WADI) data sets [71].

In addition to the SWaT data set, Yuan et al. [137] also make use of a data set collected from a gas pipeline data testbed, which involves sensors for gas pressure as well as pumps as actuators [84]. The authors also employ a GAN for the detection of anomalous behavior patterns in the data set, which are caused by multiple attacks such as command injection attacks that aim to interrupt the process flow. Filonov et al. [38] present a data set that they collect from a simulation of a gasoil heating loop, where adverse changes of temperatures or frequencies are injected as attacks. In the same publication, they show how a LSTM RNN is able to predict faults in the collected multivariate time-series data.

Seong et al. [104] point out that a majority of approaches focus on point outliers, i.e., a data point that deviates from others that are in close temporal proximity, but ignore subsequence anomalies, i.e., collective anomalies that deviate from normally repeating patterns. To address this gap, they experiment with a GRU RNN and evaluate their approach on four different cyber-physical proccesses, including boiler, turbine, water treatment, and a simulation that combines these processes. Aforementioned approaches are either unsupervised or semi-supervised, meaning that the detection of anomalous system states is only based on the deviation of the system behavior during these periods of time in comparison to the normal behavior used for training.

Mahmoud et al. [80] on the other hand propose to run supervised and unsupervised detection in parallel to combine the advantages of both strategies, which they demonstrate in a data set collected from a water distribution system. They state that supervised methods such as support vector machines (SVM) and k-nearest-neighbors enable fast and adaptive detection of anomalous states, while unsupervised methods such as isolation forests are useful for accurate confirmation of attacks.

Stojanovic et al. [114] propose a deep learning based method for detection of attacks in water distribution environment, utilising BATADAL [116], a collection of sensor readings. This paper proves the applicability of AutoEncoders in this environment and outperforms state-of-the-art approaches.

5.5.2 Categorical system statuses

While most of the data sets used by previously discussed approaches involve categorical or binary features, they are usually not explicitly modeled as such. To address this aspect, Hong et al. [49] propose an approach based on Hidden Markov Models to model system behavior with latent states and detects anomalous transitions between states as potential attacks. They test the detection algorithm in two different test environments, one where communication bus messages with purely categorical features are analyzed for noise and DoS attacks, and another one where continuous sensor and actuator values are logged from a consumer robot.

5.6 VANET detection

Attack detection in vehicular ad-hoc networks (VANET) is related to ICS intrusion detection in the sense that sensor measurements play an essential role in data collection. However, due to the fact that the monitored systems are vehicles rather than industrial machines, the collected data features as well as attack vectors are rather different to other application domains. For example, Yu et al. [136] collect a multivariate time-series from logs that specify speed and acceleration of a given vehicle as well as distance, count, average speed, and average acceleration of other nearby vehicles measured in time windows. They train an LSTM RNN in a supervised manner to classify time-series into four classes, namely normal traffic, accident, poor road condition, and congestion. This prediction is used to verify the authenticity of messages received from other vehicles and specifically to identify false emergency messages purposefully sent out by one or even multiple collaborating adversaries.

Suda et al. [115] also employ RNNs to detect data falsification in messages sent within VANETs, but additionally consider flood attacks, i.e., transmission of a large number of messages in the network. The authors point out that one of the main issues with conventional detection is the need to set a suitable threshold, which is often difficult to determine. To address this matter, they design the output layer of their network to comprise two nodes—one for normal, the other one for attacks—to enable detection without the need for thresholds.

5.7 Alert filtering and prioritization

Alerts origin from intrusion detection systems, such as the anomaly-based approaches outlined in the previous sections. In addition, signature-based approaches that rely on simple pattern- or string-matching are widely used to recognize known attacks that create specific artifacts in log data, e.g., the presence of a domain name that is known to relate to malicious activities. In Security Operations Centers (SOC), alerts are frequently collected from many devices, e.g., all servers and user machines within an enterprise, and involve a significant number of false positives. As a result, security operators often need to deal with an endless stream of highvolume alerts; a situation often referred to as alert flooding that is known to cause fatigue [5].

Researchers have therefore proposed methods to filter relevant alerts that stand out from regularly occurring alerts that are assumed to be echoes of normal system use [128]. When alert counts are considered, this means that operators should be notified about abrupt changes of the resulting time-series, while smooth changes are considered to relate to noise of normal activities or concept drift. Note that from a methodological point of view, this is similar to how anomalies that possibly relate to attacks are identified within otherwise normal events as outlined in the previous sections.

5.7.1 Conventional time-series analysis

Viinikka et al. [127–129] propose several approaches based on conventional time-series modeling that aim to deal with high numbers of irrelevant alerts from the signature-based Snort intrusion detection system collected over multiple weeks. In alignment with the data they use for evaluation, they assume that there is a continuous flow of alerts with somewhat steady frequency of incoming alerts, and that only deviations of the normal flow behaviour should be reported to operators. Alert frequency is measured in time windows with various lengths from 1 min to 1 h. The authors state that flows with somewhat constant frequencies often have machine-related origins and that periodic patterns are often the result from human activity, while random movements of the time-series without clear structure are generally difficult to explain. To obtain the alerts corresponding to these random patterns, they experiment with Exponentially Weighted Moving Average (EWMA) models that are useful to compute a smoothed estimation for the mean and predict a confidence interval for new measurements [127]. Specifically, they decide that the process generating alerts has significantly changed when alert frequencies exceed the critical values of confidence intervals.

In a follow-up publication [129], the authors abandon EWMA as they found it insufficient to adequately cope with periodic patterns and accurately detect non-drastic changes within the alert flow. Instead, they first remove the trend and periodicity of the time-series through differencing with a lag of 1 and a lag that corresponds to the length of the period. Subsequently, they use an autoregressive (AR) model to remove the stationary structure of the time-series and obtain the noise alerts. They acknowledge that some of these alerts are attributed to normally occurring artifacts or model deficiencies, and thus only report alerts corresponding to sufficiently strong deviations to normal occurrence frequencies. In their later work [128], the same authors improve on their method and apply non-stationary autoregressive (NAR) models rather than AR models and utilize Kalman filters for parameter estimations. Again, confidence intervals are used to determine whether time windows of alert frequencies are anomalous or not. Their findings indicate that the non-stationary model outperforms their previous approaches and that the selection of the time window length for counting the alerts is crucial.

5.7.2 Neural networks

Zhao et al. [142] show that neural networks are also an effective method for alert ranking and prioritization. Other than the approaches by Viinikka et al. [127–129] who only make use of alert frequencies, they consider additional features of time-series such as seasonality, severity, and inter-arrival times, as well as metrics from auxiliary sources such as response time or CPU utilization. They feed the resulting feature vectors into an LSTM RNN alongside some textual attributes of alerts for the purpose of anomaly detection. However, rather than a binary decision between anomalous and normal classes, they design their approach to compute anomaly scores which they deem superior due to the facts that they enable prioritization and handle class imbalances more effectively.

5.8 Alert forecasting and trends

Even though the previous sections show that prediction of upcoming values and estimations for confidence intervals for time-series are elementary for anomaly detection, the approaches themselves are reactive, that is, they only report anomalies after some incident occurred. Cyber security analysts, however, have strong interest in proactive security that allows to prevent or mitigate incidents before they occur. Therefore, time-series analysis is applied to forecast the number of alerts in future time intervals as well as their occurrence patterns [50].

5.8.1 Prediction of normal behavior

Kohlrausch et al. [61] analyze time-series that occur in Computer Security Incident Response Teams (CSIRT) or SOCs. They differentiate between time-series based on event counts as well as numeric indicators, such as the mean recovery time of incidents. They select ARIMA as a suitable model for analysis of these types of time-series due to the abilities to predict future values, quantitatively validate the accuracy of the model, generate stationary baselines for comparisons, and detect trends over time. As part of their evaluation, they fit an ARIMA model to daily security events (i.e., alerts) collected from a threat intelligence platform and use it for forecasting the number of alerts as well as outlier detection.

Similarly, Husák et al. [50] gather 1 week of alert occurrences in count intervals of 5, 15, 30, and 60 min from an alert sharing platform, which they model with ARIMA and Error-Trend-Seasonality (ETS) models to produce up to 10-

step-ahead forecasts, where each step corresponds to a time interval. The authors point out that one of the downsides of this kind of predictive analytics is that only the expected number of attacks is predicted, but there is no detailed information on the attacks themselves. Meanwhile, Ansari et al. [8] use GRU RNNs to understand dependencies in security alert sequences through sequence modeling and predict future actions of attackers.

5.8.2 Prediction of alert bursts

Other approaches aim to predict the occurrence of burst attacks, i.e., attacks that involve a high number of events or alerts such as brute force attacks that are commonly used for password guessing. The goal of Silva et al. [108] is to use predictive analytics to detect the start of a brute force attack, not when it is already active as this is often too late. Their idea is to analyze the total number of brute force attacks and apply multiple Moving Average (MA) models simultaneously on this time-series to predict burst attacks. Specifically, they recognize that upcoming attacks affect Moving Average models with different observation periods in such a way that their predicted values intersect at some point in time, which can be used as an indicator for imminent burst attacks.

Burst attacks are also the focus on the work by Bakdash et al. [13], who make use of malware occurrences that are detected and manually verified by analysts. They claim that this data is superior to IDS alerts or honeypot data that are commonly used for this purpose, because it involves fewer false positives and is generally of higher quality. Their approach uses Bayesian State Space Model (BSSM) for forecasting and ingests weekly event counts from a period of over 7 years. The authors find that attacks sometimes appear in systematic ways, which allows forecasting; however, bursts are generally difficult to predict beforehand as they can emerge without any prior indicators and can thus often only be detected after the fact.

5.9 Cyber situational awareness

This section shifts the focus from the low-level technical input data discussed in the previous sections, such as raw log events or intrusion detection alerts, to more high-level reports that enable strategic decision making and risk estimation rather than the detection of anomalies. Since the analyzed data does not just origin from a single component or organization but instead from a wide range of distinct organizations, the results of this kind of research are relevant to wide audiences and enable assessment of the state of cyber security at large. For example, decision makers could be interested to assess their current risk of becoming subject to cyber attack to react accordingly in advance. To this end, analysts need to deduce whether the number or severity of attacks on certain types of organizations is currently increasing or expected to do so in the future, which is feasible by applying time-series models to cyber incidents from the past and extrapolating to the future.

5.9.1 Prediction of the number of incidents

Werner et al. [132] aim to forecast the number of attacks occurring within a time window of one day solely based on the number of daily attacks in the past. For their experiments, they use 10.5 months of attack counts for different types of attacks, such as DoS, attacks over email, malicious URLs, and attacks on Internet-facing-devices. They recognize that their database is far from comprehensive—only 613 attack reports are available for this time period—but assume that this is a sufficiently large sample to represent the global threat landscape. Their results suggest that the ARIMA model they apply on the time-series is not able to predict exact values due to high fluctuations in the data; however, the authors claim that it is able to capture and predict the overall trends.

Zhang et al. [141] also predict the future number of cyber attacks, but do so on a shorter time scale than Werner et al. [132]. Specifically, they use both hourly and daily event count vectors collected from a globally dispersed set of honeypots. Using that data set, they point out the peculiar distribution of attack occurrences over time, which involves high numbers of attacks in short time intervals. To ensure that their forecasts of attack numbers integrates this aspect, they rely on deep learning to predict the mean values, while Extreme Value Theory (EVT) is used to estimate the short-term bursts.

Samia et al. [100] use an even shorter time window for event counting. They count the number of attacks in time intervals of 5-30 minutes in a data set that spans over one week and contains around 144 thousand entries. Each incident also includes additional features such as the type of attack, industry of the attacked organization, and the countries where attackers and victims reside, which the authors use to generate distinct time-series for specific countries and attack types. Interestingly, they use a very short period of only four consecutive data points to predict the upcoming value. Their results suggest that approaches based on machine learning outperform statistical methods such as ARIMA; however, they also come to the conclusion that prediction is largely limited to overall trends while exact values are difficult to forecast.

5.9.2 Prediction of cyber risks

Rather than time-series of attack counts, Zängerle et al. [138] generate time-series of yearly financial losses that organizations suffered from cyber attacks based on publicly disclosed loss events in the financial sector. They point out that the resulting time-series are sparse since many companies did not report any losses in several years. To predict the probability of attack occurrence as well as the financial impact on organizations, the authors suggest to employ separate models for different aspects of the prediction; specifically, they use a marginal model for the loss, logistic regression for the attack frequency, Extreme Value Theory (EVT) for severity, and a copula structure for the seasonal trend. Their results suggest that financial losses caused by cyber attacks are often not as severe as assumed and do not reach the scales of other risk events that are not related to security incidents. Moreover, the cyber risks appear different across industrial fields, which thus need to be analyzed separately.

Dzhamtyrova et al. [32] also aim to predict cyber risks through the inter-arrival times and sizes of data breaches. Their data set comprises dates and the number of affected accounts from more than 9, 000 breaches collected in course of 14 years. Other than aforementioned approaches, they process the inter-arrival times and loss sizes directly rather than aggregating them within time windows, resulting in fine-grained time-series. To predict the expected confidence intervals for these time-series, they adopt quantile AR (QAR) and competitive quantile AR (CQAR) models that are capable of modeling both occurrences of cyber incidents and their losses as stochastic processes and allow to evaluate them for arbitrary significance levels. Eling et al. [33] point out that the data set used by Dzhamtyrova et al. [32] only entails the number of affected accounts, but not the financial loss; therefore, they evaluate their method with two additional data sets where information on financial losses for cyber incidents is present. The authors argue that exact loss amounts are generally only known some time—possibly months—after incidents occur, and that this delay introduces a bias in the data which needs to be compensated for. To address this issue, they explicitly integrate delay times as part of their models, which assume that occurrence numbers and scales of loss events follow statistical distributions.

5.9.3 OSINT-based incident prediction

The main idea of incident prediction based on Open-Source Intelligence (OSINT) is to leverage information from unconventional data sources such as news feeds or posts in social media platforms and link them to cyber incidents. Thereby, it is sometimes possible to forecast upcoming cyber attacks that are launched as a reaction to certain real-world events covered by new stories [90]. The obvious problems encountered in such an endeavour include the often highly unstructured nature of the data, which often involves natural language, as well as a high fraction of noise in the data, i.e., events without relevance to real-world incidents or cyber security in general. For example, Okutan et al. [90] obtain time-series from a global event database, news feeds, Twitter posts, sentiment signals, etc, as well as a non-public time-series with binary observations of attack types. They forecast the expected number of attacks for any given day and correlate the two time-series to find suitable aggregation time window sizes as well as delays, which allow them to train a Bayesian classifier to forecast future occurrences of certain attack types.

Lande et al. [68] consider event mentions in sources such as news feeds or Internet blogs to create event count vectors. For their experiments, they obtain 1 year of data and aim to predict the number of events in the last month using the first 11 months as training data. They compare four different time-series methods, of which ARIMA turns out to yield the best predictive performance but also require parameter fine-tuning, while LSTM RNN is more simple to apply at the cost of a lower accuracy of the prediction.

5.10 Vulnerability analysis

Vulnerabilities of software or cyber-physical systems pose a high risk to organizations and individuals. Specifically, zero-day vulnerabilities that are not known by vendors or operators of affected systems have often been the entry point of some of the largest cyber incidents in the past. Unfortunately, due to the ever-growing complexity of software, the number of vulnerabilities has continuously increased in the past and is expected to continue to do so in the future [96, 135]. For strategic decision making that concerns the security level of software products and components it is thus vital to estimate whether or how many vulnerabilities are expected to be reported in certain time intervals [58]. Security experts therefore leverage vulnerability data from large and open databases such as the National Vulnerability Database (NVD) to analyze the patterns of vulnerability appearances over time and extrapolate trends into the future. Note that other than the approaches of the previous section that considered the number and severity of attacks, vulnerability analysis primarily focuses on the first appearance of a vulnerability rather than the instances where these vulnerabilities are exploited.

Pokhrel et al. [96] focus on vulnerabilities in widely used operating systems, specifically, Windows 7, Mac OS X, and Linux. They collect monthly counts of reported vulnerabilities over up to 15 years and discuss how the consequences of certain cyber incidents manifest in the data. They point out that even though the time-series are affected by seemingly random fluctuations and there are no visually apparent evidences for seasonality, certain time periods show some

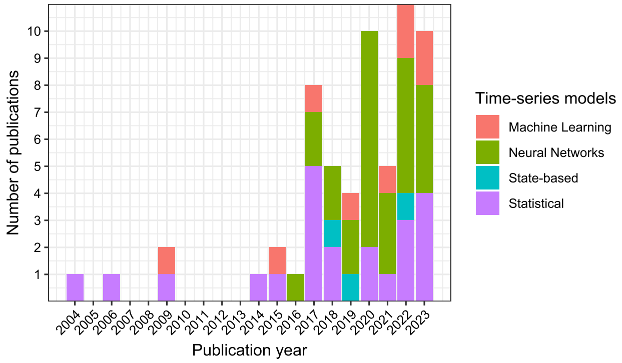


Fig. 2 Number of publications per year separated by classes of time-series models

sort of increasing or decreasing trends. They experiment with ARIMA, neural networks, and Support Vector Machines (SVM) to model each of the three time-series and predict future appearances of vulnerabilities. In addition to operating systems, Yasasin et al. [135] consider the vulnerabilities of software applications such as various Internet browsers. Similar to Pokhrel et al. [96], they consider multiple years of monthly vulnerability counts and predict up to three months ahead. Among the many methods they experiment with, Croston's method and ARIMA turn out to be the most adequate approaches to model the time-series. Specifically, they are more capable of handling time-series that contain many zero values than other models such as neural networks, which is an important aspect in vulnerability analysis since some software have no reported vulnerabilities for several months at a time. Kalouptsoglou et al. [58] experiment with a similar selection of vulnerability time-series and an even more diverse set of modeling techniques. They found that the there is no optimal model for all time series; overall, their results suggest that statistical models usually outperform neural networks when it comes to fitting to the time-series, while approaches from both categories are on-par regarding the ability to predict up to 24 months ahead.

6 Time-series models

The literature review presented in the previous section highlighted the diversity of time-series methods applied on security data. This section thus summarizes the main techniques for modeling time-series that we encountered during our review. We identify four different classes of models, each comprising several sub-classes, that we use to structure this section: statistical models, conventional machine learning, neural networks, and state-based models. Figure 2 shows how many publications using each class of time-series models appeared per year. It is apparent that statistical models and neural networks are the two most frequently used techniques for time-series analysis; however, all classes of models are still employed by 2022, indicating the need to consider a wide range of models when dealing with detection and forecasting of security time-series.

6.1 Statistical models

This section summarizes statistical models for time-series analysis, including exponential smoothing, autoregressive and moving average models, and Croston models. In addition, we provide some insights on correlation analysis and state methods to capture the seasonality of time-series.

6.1.1 Exponential smoothing

Exponential smoothing is a local method for time-series prediction and has multiple implementations: Single exponential smoothing (SES) uses weighted averages of preceding observations for forecasting, where weights decrease for observations that lie further in the past. A parameter α is used to adjust the weights, where smaller values of α correspond to higher weights to observations that are in the distant past, and vice versa. The prediction depends on a linear parameter l_t (cf. Eq. 2); due to this definition, every value predicted for an arbitrary horizon h into the future is identical (cf. Eq. 1) [50, 58, 86, 108, 119, 127, 135].

$$\hat{y}_{t+h} = \hat{y}_{t+1} = l_t \tag{1}$$

$$l_t = \alpha y_t + (1-\alpha) l_{t-1} \tag{2}$$

Double exponential smoothing (DES) extends SES with a linear trend component. To this end, the slope of the time-series (cf. Eq. 5) is added to the equations for the level (cf. Eq. 4 and the prediction (cf. Eq. 3), where parameter β is used to weigh the slope between two successive observations [135].

$$\hat{y}_{t+h} = l_t + h b_t \tag{3}$$

$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{4}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta) b_{t-1} \tag{5}$$

Triple exponential smoothing (TES), which is also known as Holt-Winters exponential smoothing, extends DES with a seasonal component. To this end, another parameter γ is introduced to weigh the seasonal effect over the m most recent time periods (cf. Eq. 9) and added to the level (cf. Eq. 7) and the prediction (cf. Eq. 6), where $h_m = ((h-m) \mod m)+1$ ensures that the correct season is used for computing the estimates [58, 135].

$$\hat{y}_{t+h} = l_t + h b_t + s_{t+h_m-m} \tag{6}$$

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{7}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{8}$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{9}$$

6.1.2 Autoregressive (AR) models

The most simple models have global characteristics, i.e., even the oldest data point influences the prediction. The autoregressive (AR) model is a counter-design to the global approach: according to this model, the next value depends only on the p-last values. Values further back in time are not taken into account, where p is called the order of the model. Moreover, the AR model consist of model parameters $\phi_1, ..., \phi_p$ and an error term ϵ_t as stated in Eq. 10 [54].

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \epsilon_t \tag{10}$$

Note that once the parameters to adequately model a time-series have been identified, it is possible to leverage these parameters to extrapolate over the most recent time point and produce forecasts for future observations $\hat{y}_{t+h}$. The model can further be used to detect anomalies by checking deviating values against the usual variance of the error term. Viinikka et al. [128] use a non-stationary AR model (NAR) with time-varying parameters to allow the model to adapt to changes of normal behavior. Other variations are the quantile autoregressive model (QAR), which models each quantile of time-series values with a separate autoregressive process, and competitive quantile autoregressive model (CQAR), which allows incremental updates and does not require training data [32].

The AR(1) model, i.e. the autoregressive model with order 1, is commonly called a Markov process and is written mathematically as in Eq. 11, where μ is the mean level of the process, ϕ_1 is the AR parameter, ε_t is the white noise term at time t that is identically independently distributed with a mean of 0 and a finite variance [48].

$$y_t - \mu = \phi_1(y_{t-1} - \mu) + \varepsilon_t \tag{11}$$

In the literature, the notation of the autoregressive model is not always the same—in some cases the mean μ of the series is explicitly specified, in others it is not. Another significant difference concerns the usage of the backward shift operator B, which is defined by Eqs. 12 and 13 with a positive integer k.

$$\mathrm{B} y_t = y_{t-1} \tag{12}$$

$$\mathrm{B}^k y_t = y_{t-k} \tag{13}$$

By using the backward shift operator B, and by treating B as an algebraic operator and factoring, the Markov process in Eq. (11) becomes the expression in Eq. 14, where $\mathrm{B}\mu = \mu$ since the mean level is a constant at all times.

$$(1 - \phi_1 \mathrm{B})\,(y_t - \mu) = \varepsilon_t \tag{14}$$

The AR model only depends on a few previous values to compute the next step. Therefore, short-term effects can be modeled in an easier way, but the global structure of the model is not obvious. Nevertheless, the global structure of the AR model can be determined [124].

6.1.3 Autoregressive moving average (ARMA)

The autoregressive model described above determines the value at time step t based on the p-last values (cf. Eq. 15).

$$y_t = \phi_1 \cdot y_{t-1} + \phi_2 \cdot y_{t-2} + \ldots + \phi_p \cdot y_{t-p} + \varepsilon_t \tag{15}$$

The inaccuracies of the model and external influences are modeled with statistical noise ε_i. However, if these influences not only affect one point in time, but have an impact over several time steps, this must be taken into account in the model. The autoregressive moving average (ARMA) model follows exactly this approach by extending the autoregressive model.

The autoregressive moving average model describes a process using a linear combination of several previous values (cf. Eq. 16) [17].

$$y_t = \mu + \varepsilon_t + \sum_{i=1}^{p} \phi_i \cdot y_{t-i} + \sum_{j=1}^{q} \psi_j \cdot \varepsilon_{t-j} \tag{16}$$

The signal y_t to be modeled is composed of

- a constant μ describing the base level,
- a noise term ε_t to model inaccuracies,
- a weighted sum of the p preceding signal values, and
- a weighted moving average of noise terms $\varepsilon_{t-1}, ..., \varepsilon_{t-q}$.

This model is abbreviated ARMA(p, q), where p and q indicate the autoregressive and moving-average order of the process, respectively. Special ARMA models with $p = 0$ or $q = 0$ are the autoregressive model and the moving average (MA) model.

The parameters of an ARMA model can be determined in many ways. The approach to the solution can be numerical or stochastic-statistical; i.e. in most cases an algorithm according to Yule–Walker [40], maximum likelihood [19] or two-step-regression is implemented. Each approach has its justification; however, they may lead to different results.

In the reviewed literature, specialized versions of ARMA models are employed. For example, Dzhamtyrova et al. [32]

use Generalized AutoRegressive Conditional Heteroscedasticity (ARMA-GARCH) models, where the GARCH part is used for modeling the voltatility of the time-series, which depict breach sizes in their application case. Ara et al. [9] use a βARMA model, which is specifically developed for a random variable with beta distribution. More precisely, they use a bivariate βARMA model for multiple variables and leverage a copula to create a joint distribution for both variables.

6.1.4 Autoregressive integrated moving average (ARIMA)

The autoregressive moving average model is the basis for various, extended models. An obvious extension is to convert scalar time series to vector-based time series; formally, this difference is relatively small. Other extensions address effects that are difficult to model without the corresponding extensions.

If a time series does not describe “fluctuations” around a value, but a summed series, then this behavior can rather be described with an autoregressive *integrated* moving average (ARIMA) model. The summation (respectively the integration) is considered in this particular model. The integrating aspect can be removed by difference transformation, so that the transformed model corresponds to an ARMA model.

In detail, the transformation takes the initial values (cf. Eq. 17) and transforms them into a sequence of differences (cf. Eq. 18).

$$y_{-N+1},\ y_{-N+2},\ y_{-N+3},\ \dots,\ y_{-2}, y_{-1}, y_0, \tag{17}$$

$$\underbrace{y_{-N+2}-y_{-N+1}}_{z_{-N+2}},\ \underbrace{y_{-N+3}-y_{-N+2}}_{z_{-N+3}},\ \dots,\ \underbrace{y_{-1}-y_{-2}}_{z_{-1}},\ \underbrace{y_0-y_{-1}}_{z_0} \tag{18}$$

An ARMA(p, q) model is then created for the sequence of z_t, which can be used to determine a forecast z_1, z_2, z_3, etc. Using the inverse differences, the z_t can be used to determine the corresponding y_t (cf. Eq. 21).

$$y_1 = y_0 + z_1 \tag{19}$$

$$y_2 = y_1 + z_2 = y_0 + z_1 + z_2 \tag{20}$$

$$\vdots$$

$$y_k = y_{k-1} + z_k = y_0 + \sum_{i=1}^{k} z_i \tag{21}$$

This difference scheme is a so-called ARIMA$(p, 1, q)$ model. Differences of higher order (differences of differences) lead to an ARIMA(p, d, q) model. The integer d corresponds to the order of the differences. First order differences are computed as in Eq. 22 whereas higher order differences are defined recursively as in Eq. 23.

$$\Delta^1 y_i = \Delta y_i = y_i - y_{i-1} \tag{22}$$

$$\Delta^m y_i = \Delta^{m-1} y_i - \Delta^{m-1} y_{i-1} \tag{23}$$

To differentiate between AR and MA models and determine model parameters, authors usually make use of the autocorrelation function (ACF) and partial autocorrelation function (PACF) [107, 142]. Moreover, authors use the Akaka’s information criterion (AIC), the AIC corrected (AICc) criteron, and the Bayesian information criterion (BIC) [9, 32]. While this is often carried out visually, some authors also rely on autofitting methods that automatically find suitable parameters [50].

ARIMA models are among the most commonly used methods for time-series analysis in the reviewed literature and applied on various data sources for prediction and anomaly detection, including log event counts [60, 67], generic event counts [68, 90], alert counts [50, 61], attack counts [100, 132], vulnerability counts [96, 135], and numeric event parameters [86, 107]. Even IBM’s QRadar[4] and DataDog[5] rely on ARIMA models for log analysis and intrusion detection.

6.1.5 Seasonal models

If a time series has a seasonal pattern, it is not sensible to describe the complete sequence of values y_t using one model. For example, in a monthly time series it may make more sense to base the forecast of next month’s value on the previous year’s value than to base it on the previous month’s value. In detail, a sequence of values is broken down into several sequences (cf. Eq. 24) and each subsequence y_t, y_{t+k}, y_{t+2k}, y_{t+3k}, … handled individually using a non-seasonal model (ARIMA, ARMA, …).

$$\begin{array}{c|c|c|c|c} y_1 & y_2 & y_3 & \dots & y_k \\ y_{k+1} & y_{k+2} & y_{k+3} & \dots & y_{2k} \\ y_{2k+1} & y_{2k+2} & y_{2k+3} & \dots & \dots \end{array} \tag{24}$$

The value k is called *seasonal lag* and describes the number of intermediate values until a seasonally equal value occurs again. Determining the seasonal lag is often based on domain knowledge in the reviewed literature. One exception is the work by Wu et al. [133], who make use of a framework that relies on Fourier series to extract seasonal terms.

[4] https://www.ibm.com/qradar.

[5] https://docs.datadoghq.com/.

6.1.6 Croston models

One of the issues with aforementioned models is that they are not designed to work with time-series that contain a high fraction of zeros, which may be the case when deriving time-series from unevenly distributed count data. An alternative method that tackles this problem is provided by Croston models, which decompose the original time-series into two new time-series, one without the zero values (cf. Eq. 26) and another one that captures the durations between intervals comprising only zeros (see. Eq. 27). Then, single exponential smoothing is applied on each of these two time-series separately. Finally, the estimate of the mean of non-zero values is computed as stated in Eq. 25 [58, 135].

$$\hat{y}_{t+h} = \frac{\hat{z}_{t+h}}{\hat{v}_{t+h}} \tag{25}$$

$$\hat{z}_{t+h} = \begin{cases} z_t & \text{if } y_t = 0 \\ \alpha y_t + (1-\alpha) z_t & \text{if } y_t \neq 0 \end{cases} \tag{26}$$

$$\hat{v}_{t+h} = \begin{cases} v_t & \text{if } y_t = 0 \\ \alpha y_t + (1-\alpha) \hat{y}_t & \text{if } y_t \neq 0 \end{cases} \tag{27}$$

6.2 Conventional machine learning

Other than aforementioned statistical models that are specifically designed to handle time-series, most machine learning techniques used in the reviewed literature are originally not conceptualized for such data. Instead, most machine learning techniques usually process independent data samples and neglect order or temporal dependencies. Nonetheless, some authors have managed to adapt the techniques or integrate temporal aspects in the data samples in such a way that the models are able to capture time-dependent information. In the following, we go through different machine learning techniques and outline how they were employed with time-series.

6.2.1 N-grams

N-grams are a common modeling technique in natural language processing, where contiguous sequences of words are used to predict the probabilities of subsequent words. The same idea can be applied to time-series, where similar subsequences of values are located in the time-series to estimate the subsequent value, following the assumption that the same pattern that occurred in the past repeats [68]. The authors find that the limited number of past values deteriorates the performance of the prediction and point out that models based on n-grams are generally criticised for not being able to incorporate long-term dependencies of the data.

6.2.2 Decision trees

There are three models based on decision trees in the reviewed literature. First, isolation forests operate by partitioning data into tree structures using randomly selected features. The idea is that anomalous instances have more distinguishable values in some of the features and thus yield shorter paths in the resulting trees, compared to normal instances that are less likely to be isolated and thus yield longer paths [7, 80]. Second, random forests that similarly rely on ensembles of decision trees but use labeled data for supervised training [80, 89, 100]. Third, Extreme Gradient Boosting (XGBoost) is another supervised approach that uses decision trees for ensemble learning and is a specifically popular choice in machine learning due to its performance-driven nature.

6.2.3 K-nearest neighbors

The K-nearest Neighbors (KNN) supervised classification technique uses majority voting of the closest instances (e.g., in a numeric space) to determine the class of an unknown instance, e.g., whether it is normal or anomalous. Thereby, instances that are further away from the instance in question may receive lower weights [80]. The K-nearest neighbor regression (KNR) is a related technique that uses continuous values rather than classes of the closest instances to forecast numeric properties of instances, for example, by taking the average of the values in the K-nearest neighbors [100]. Since the approach is designed to handle numeric data, it is primarily used with sensor values collected from cyber-physical systems.

6.2.4 Support vector machines

Support Vector Machines (SVM) is a class of supervised learning models capable of performing linear and non-linear classification by drawing boundaries between instances that are mapped to high-dimensional feature spaces [80]. One-class SVM is a semi-supervised alternative that is trained on normal data and fits a boundary that encompasses these instances. Then, instances with unknown class are determined as normal if they are within these boundaries, and anomalous otherwise. Note that authors do not necessarily feed time-series directly in the SVM, but properties such as percentiles, standard deviations, skew, seasonality, etc [119]. Support Vector Regression (SVR) is based on SVMs and used for the prediction of continuous values, including time-series [96, 100].

6.2.5 Local outlier factor

Local Outlier Factor (LOF) is a density-based technique for anomaly detection that is based on the local density of an instance in relation to the densities of its neighbors. LOF has been applied on time-series by dividing the data sets into time windows of numeric and one-hot encoded categorical features [7].

6.2.6 Clustering

Clustering groups instances based on a similarity metric, often with the goal to find outliers that do not fit into any cluster. For example, Du et al. [31] cluster sub-sequences of event count vectors and compute an anomaly score based on the similarities. Jain et al. [56], on the other hand, pursue temporal clustering of log data where the similarity of event count vectors is measured with the Jaccard metric for the purpose of correlation analysis.

6.3 Neural networks

In recent years, neural networks have gained high popularity in the research domains of image classification as well as natural language processing. Given that some neural network architectures have been developed specifically for sequential data, it stands to reason to apply them for the purpose of time-series analysis. Other types of neural networks are capable of operating in semi-supervised manner and thus support anomaly detection. Some authors that conduct comparative studies find that methods based on neural networks slightly outperform conventional machine learning methods and by far outperform statistical models [100]. In the following, we summarize the most common neural network architectures and their application in the reviewed literature.

6.3.1 Feedforward neural network

The architecture of feedforward neural networks comprises multiple layers with varying numbers of nodes, in particular, an input layer that ingests the raw data, one or more hidden layers that connect nodes across each layer, and an output layer [101, 102]. Several authors forecast time-series using such a neural network with a single hidden layer [96, 135]. Yasasin et al. [135] point out that feedforward neural networks are more suitable to handle time-series with many zeros than other models.

6.3.2 Recurrent neural networks

Other than feedforward neural networks, the architecture of Recurrent Neural Networks (RNN) involves loops so that the output of some nodes affects their own input. Since these feedback mechanisms are designed to retain certain states of the model over time, RNNs are inherently capable of learning sequential information from raw data and thus a natural choice for processing time-series [100–102]. Accordingly, RNNs present one of the most used techniques for time-series analysis in the reviewed literature. Thereby, the most common implementation is the Long Short-Term Memory (LSTM) RNN, which enable long-term storage of states and comprise input, output, and forget gates [38, 68, 119]. Gated Recurrent Units (GRU) are neural network cells that only use update and reset gates. They are generally considered as alternatives to LSTM RNNs in cases where high computational efficiency is required. Some authors compare different architectures, e.g., Zhang et al. [141] experiment with LSTM, GRU, and multiplicative LSTM to learn patterns from event counts, and Kalouptsoglou et al. [58] benchmarks various statistical methods and neural network architectures.

Interestingly, there are many different ways how data is fed into and derived from RNNs. Samia et al. [100] design the input layer of their RNN to have the same dimension as the length of the time-series sub-sequence they want to analyze. The output layer, on the other hand, consists of a single node that predicts the next value in the time-series.

RNNs are also used with categorical event data, where sequences of event types are fed into the neural network. Čeponis et al. [21] then use a binary output for attack detection, while Meng et al. [81] try to predict the most probable next event types in the sequence and thereby detect anomalies. Wang et al. [130] use a different strategy and predict the time-of-day where attacks start or stop.

Several approaches use multivariate data as input to the RNN, for example, Seong et al. [104] use 89 observations of 79 numeric features to predict the upcoming observation. Yu et al. [136] use a multivariate time-series of road traffic measurements and use four output nodes corresponding to four traffic situations that they consider relevant. Suda et al. [115] use two output nodes and use the difference between their output values as a measure for attack detection.

The work of Wu et al. [133] specifically focuses on multi-seasonality of time-series data. For this reason, the authors do not only provide windows of multivariate time-series as input to the neural network, but also their seasonality. The output of the neural network corresponds to future time windows, which they convert to an anomaly score. Zhao et al. [142] use time windows of event counts and additionally use features such as seasonality and inter-arrival times as input. They use the prediction error of the network output as a measure for the degree of an anomaly. Finally, Anastasiadis et al. [7] combine an RNN with an AutoEncoder neural network to conduct anomaly detection.

6.3.3 Generative adversarial networks

Generative Adversarial Networks (GAN) comprise two neural networks that compete against each other. The generator is trained to produce new samples that resembles training data, while the discriminator is trained to determine whether the generated data is taken from the training data, which is in turn used to improve the generator. In the reviewed publications, these neural networks are realized with RNNs [101, 102].

GANs have been used to generate missing data that is suitable for interpolation [137]. Moreover, GANs enable anomaly detection since the discriminator is trained to distinguish fake data from real samples anyway and the generator can be used to derive an anomaly score from the reconstruction error of the test samples [70, 71]. Khoshnevisan et al. [60] propose to use a Robust Seasonal Multivariate Generative Adversarial Network (RSM-GAN) that adopts convolutional LSTM RNNs and attention mechanisms, which they specifically select to handle seasonal and noisy data.

6.3.4 Convolutional neural networks

Convolutional Neural Networks (CNN) extend on the architecture of feedforward neural networks by adding convolutional and pooling layers, which facilitate dimension reduction and allow the network to capture more abstract features [101, 102]. For event data, CNNs have been applied on univariate time-series of count vectors [58], sequences of event types [21], and word-embedding of alert messages [89].

6.3.5 Attention mechanisms

Attention mechanisms are used to assign higher weighs to relevant features than to irrelevant ones [101, 102]. Khoshnevisan et al. [60] use an attention layer to compensate for the fact that older data points may be more suitable for prediction than the most recent ones, and to incorporate holiday effects, i.e. days with unusual behavior, from the past. They then compute the reconstruction error and use it as an anomaly score for analyzed points in time.

6.4 State-based models

Most of the aforementioned time-series models assume that the input data is numeric and continuous. However, event types and parameters are often discrete or categorical, which cannot be processed directly. In the following, we summarize state-based models, which offer an alternative approach suitable for such features.

6.4.1 Hidden Markov models

When dealing with discrete time-series, for example, where observed values are integers rather than real numbers, Markov chains present a suitable method for analysis and forecasting. Naveiro et al. [86] specifically use Markov chains to estimate the transition probabilities between different states (i.e., observed values), forecast upcoming values, and compute upper and lower bounds of one-step-ahead predictions. Hong et al. [49] show that Hidden Markov Models (HMM) are also applicable on categorical features where values do not follow any order as it is the case with discrete data.

6.4.2 Bayesian state space model

Bakdash et al. [13] use the Bayesian State Space Model (BSSM) to predict the number of cyber attacks in a given time interval, based on the previously observed attack frequencies. In short, the model considers that the system moves in different states and allows to generate predictions for future observations based on the current state. The authors point out that BSSM has several advantages over statistical models, specifically, BSSM is capable of incorporating different statistical distributions, multiple sources of variability, changes of long-term trends, and non-stationary patterns in the data.

7 Evaluations of scientific approaches

Evaluations are crucial to validate and demonstrate approaches and thus a vital part of publications. However, designing experiments for sound evaluations requires critical decisions regarding the selection of data sets, methods for data pre-processing and parameter tuning, and evaluation metrics. This section therefore summarizes common evaluation strategies in the reviewed publications.

7.1 Common security data sets for time-series analysis

The quality of data sets is essential to enable sound evaluations. When data sets are flawed or in any way not representative for data that is encountered in practice, evaluation results may not generalize to real-world applications and derived insights may be misleading. Moreover, the development of analysis techniques is often driven by data properties, for example, data sets containing multiple relevant dimensions may require approaches for multivariate time-series analysis. Unfortunately, useful data sets are often difficult to find as publicly available resources. In this section, we therefore summarize open data sets used in reviewed publications. Table 4 shows an overview of data sets used to

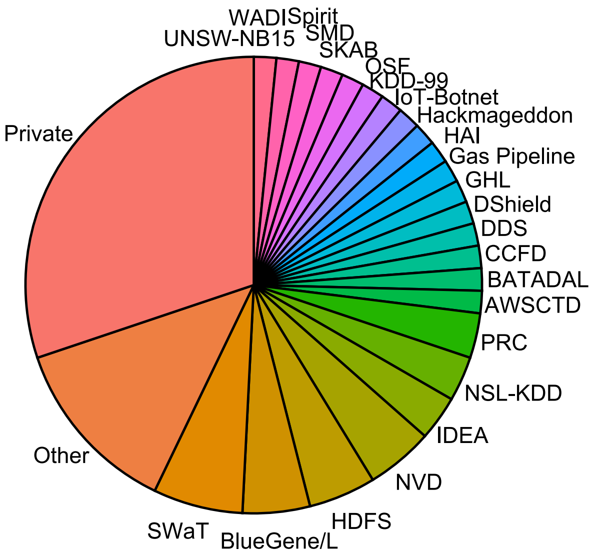


Fig. 3 Frequencies of data sets in the reviewed literature

evaluate anomaly-based approaches, where we also provide some information on relevant properties such as features, dimensions, anomaly types, and labels. Table 5 presents data sets used for analytical approaches that focus on time-series modeling rather than anomaly detection; accordingly, the data sets do not involve anomaly types or labels. Figure 3 visually summarizes the popularity of data sets, i.e., the relative number of times a data set is used in the reviewed publications. Note that "Private" refers to data sets that are not publicly available (e.g., data sets collected in productive environments that cannot be shared for privacy reasons) and "Other" refers to data sets that are either not directly linked to time-series, not available anymore, behind paywalls, or only partially available. This figure shows that a more than a quarter of all approaches are only evaluated on private data sets, which means that the presented results cannot be reproduced. In the following, we briefly describe each data set and refer to the references provided in the table for further reading.

7.1.1 Log data

Log data differs from common time-series in two crucial ways. First, log data generated by applications or operating systems generally comprises unstructured event messages, which require parsing. Accordingly, the number of dimensions, that is the number of discrete event types that these unstructured messages correspond to, depends on the granularity of log templates used for parsing. Second, log events appear at arbitrary points in time rather than fixed time intervals, which means that event occurrences are usually counted in time windows in order to obtain numeric time-series.

The Hadoop Distributed File System[6] (**HDFS**) data set that is generated by an application for storing and processing large files is such an example. The data set is a commonly used data set in log-based anomaly detection as it involves a high number of normal and anomalous event sequences, which are formed by file block identifiers present in all events. Anomalous sequences thereby relate to abnormal execution flows that indicate incorrectly processed files. Similar to HDFS, **BlueGene/L** and **Spirit** data sets (both available at the Computer Failure Data Repository[7]) contain unstructured log messages and thus require parsing and counting of events. They stem from high-performance computing clusters and involve anomalous events that correspond to system faults that are subject of detection. The Attack-caused Windows OS System Calls Traces Data set[8] (**AWSCTD**) comprises sequences of event types that are already in preprocessed form, meaning that only discrete event sequences without timestamp information are available. Other than aforementioned data sets, however, the anomalous events in the AWSCTD actually relate to security incidents rather than system faults that may also occur without malicious intent. Specifically, the authors of the AWSCTD executed several malware samples and exploits when collecting the data.

The Credit Card Fraud Detection[9] (**CCFD**) data set comprises a list of credit card transactions, of which a small fraction (492 out of 284,807) are marked as fraudulent. Aside from the time and amount of the transaction, the data set comprises 28 numeric values that are known to be the output of a principal component transformation, but not explained in any more detail by the authors for confidentiality reasons. According to the authors, the Server Machine Data set[10] (**SMD**) was collected from 28 different machines at a large Internet company. Unfortunately, there is no other information on the meaning of the 38 numeric features as well as the nature of the anomalies provided by the authors.

Several authors resort to private log data sets that are usually collected at their own premises. For example, Landauer et al. [67] use logs from the MySQL application and Wu et al. [133] use logs from a Linux server with manually labeled attacks. Another common source of log data are web applications, for example, Ara et al. [9] use web server data that they obtain from a corporation, Granlund et al. [44] use Apache Access logs, Ohana et al. [89] use logs from IBM cloud data centers, and Khoshnevisan et al. [60] use request logs.

[6] http://people.iiis.tsinghua.edu.cn/~weixu/sospdata.html.

[7] https://www.usenix.org/cfdr-data.

[8] https://github.com/DjPasco/AWSCTD.

[9] https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud.

[10] https://github.com/NetManAIOps/OmniAnomaly.

Table 4 Public data sets used to evaluate anomaly-based approaches

Name	Reference	Data source	Features	Dim.	Anomalies	Labels	Gran.	Duration
HDFS	[134]	Application logs	Event sequences	–	System faults	Binary	Event seq.	38.7 h
BlueGene/L	[91]	Application logs	Event sequences	–	System faults	Alert type	Events	214 days
Spirit	[91]	Application logs	Event sequences	–	System faults	Alert type	Events	558 days
CCFD	Kaggle	Credit card transactions	Time, transaction amount, 28 unknown numeric values	30	Frauds	Binary	Events	2 days
SWaT	[43]	Testbed for secure water treatment	Numeric sensor readings and categorical actuator states	51	Tank overflow, modify actuator	Attack type	Time-based	11 days
WADI	[3]	Testbed for water distribution systems	Numeric sensor readings and categorical actuator states	123	Tank overflow, pipe damage, modify processes, pump control	Attack type	Time-based	16 days
BATADAL	[116]	Simulation of water distribution systems	Numeric sensor readings and binary actuator states	43	Modify signals/thresholds, replay attacks, concealment	Attack type	Time-based	1 year
Gas Pipeline	[84]	Gas pipeline testbed	Time, pressure, pump, crc rate, etc.	17	Command injection, modify rates	Attack type	Events	
SMD	GitHub	Server machines	Unknown numeric features	38	Unknown	Binary	Events	5 weeks
SKAB	Kaggle	Water circulation experiments	Time, acceleration, current, pressure, temperature, etc.	10	Partly closed valves, shaft imbalance, cavitations	Binary	Events	~hours

Table 4 continued

Name	Reference	Data source	Features	Dim.	Anomalies	Labels	Gran.	Duration
AWSCTD	[20]	Windows OS system calls	Event sequences	564	Malware executions	Malware types	Event seq.	Unknown
GHL	[38]	Gasoil heating loop sim.	Continuous sensor readings and binary actuator states	19	Unauthorized changes (e.g., temperature)	Attack type	Cases	~days
UNSW-NB15	[85]	Network packet captures	IP addresses, ports, packet sizes, etc.	48	Fuzzers, backdoors, DoS, exploits, recon., shellcode, worms	Binary	Flow-based	31 h
HAI	GitHub	CPS (railways, water treatment, power plants)	Valve positions, flow rates, temperatures, etc.	80	Malfunctions of compromised sensors or controllers	Attack type	Events	7 days
DShield	DShield	SANS Internet Storm Center	Daily number of attacks (e.g., by ports)	–	–	–	–	Since 2017
KDD-99, NSL-KDD	[29, 117]	Packet and flows	Network packet captures	41	DoS, U2R, R2L, probing	Attack type	Flow-based	9 weeks
NGIDS-DS	[45]	Packet and log data	Network packet captures	16	DoS, backdoors, exploits, recon., shellcode, worms	Attack type	–	5 days
TRAbID	[126]	Packet and flows	Network packet captures	50	DoS, port scans	Attack type	–	8 h
CICIDS2017	[106]	Packet and flows	Network packet captures	80	DoS, DDoS, heartbleed, brute-force, web attacks, infiltration	Attack type	–	5 days
CSE-CIC-IDS2018	[106]	Packet and flows	Network packet captures	80	DoS, DDoS, heartbleed, brute-force, web attacks, infiltration	Attack type	–	18 days
BoT-IoT, IoT-Botnet 2020	[62, 123]	Packet and flows	Network packet captures	85	DoS, DDoS, recon., information theft	Attack type	–	–

Table 5 Public data sets used to evaluate analytical approaches

Name	Reference	Data source	Features	Dim.	Duration
IDEA	[52]	Intrusion alerts from sharing platform	Time, port, IP, protocol, alert category, etc.	–	1 week
DDS	DDS	Honeypots	Date, host, protocol, type, port, location	9	6 months
NVD	NVD	National Vulnerability Database	Date, name, status, description, references	–	Since 1999
Hackmageddon	Hackmageddon	Cyber attack statistics	Date, target, description, attack type, references, etc.	–	Since 2013
PRC	Privacy Rights	Data breaches reported in the US	Date, company, industry sector, location, attack type, breach size (affected accounts), etc.	13	Since 2005
OSF	[13]	Reports of manually detected and verified incidents	Date, weekly count, average report length	3	7 years

7.1.2 Network traffic

Prominent IDS datasets include **KDD-99**[11] and its derivative **NSL-KDD**.[12] KDD-99 holds 41 features from 5 million connection records over 7 weeks, with additional packet and flow-based formats and labeled entries for four attack types. NSL-KDD was generated by removing redundant and duplicated records in the KDD-99 and resampling the records for a more challenging classification [117]; however, the data set is currently not available on the platform where it was originally hosted. The **UNSW-NB15**[13] dataset, established in 2015, comprises 49 features from normal and malicious packet-based network traffic, covering nine attack types in packet and flow-based formats. The **NGIDS-DS**[14] dataset, created in 2016, features 7 packet-based and 9 log file-based attributes, encompassing seven attack families. Proposed in 2017, the **TRAbID**[15] dataset includes 16 scenarios capturing 30 minutes of emulated environment traffic for evaluation. The latest additions, **CICIDS2017**[16] and **CSE-CIC-IDS2018**,[17] offer network and log data with 80 features extracted from traffic, covering seven different attack scenarios each. Also, **IoT-Botnet 2020**[18] dataset presents one of the important benchmark datasets in this field. It is offered in both packet based format, and extracted features, and includes traces of different types of attacks.

7.1.3 Cyber-physical data

The Secure Water Treatment[19] (**SWaT**) data set contains sensor readings and actuator states collected from a cyber-physical test environment that represents a water treatment plant and is capable of simulating common processes such as assessment of water quality, chemical dosing, filtration, dechloration, etc. As part of their study on secure water treatment systems, the authors launch 36 attacks against the test environment that aim at damaging the infrastructure and maliciously affecting the treatment process, e.g., by causing tank overflows or altering the settings of actuators. According to the authors, the test environment where they collect the Water Distribution (**WADI**) data set is an extension of the SWaT testbed; the data is available on the same website. Again, the authors collect data both during normal operation of the plant as well as during several attack scenarios. The same team of researchers curating the SWaT and WADI data sets hosted a competition called Battle of the attack detection algorithms (**BATADAL**), as part of which a data set emerged that is hosted on the same website as SWaT. While also focusing on water treatment and attacks on the involved cyber-physical systems, the data set spans over a significantly longer duration of more than one year.

The Skoltech Anomaly Benchmark[20] (**SKAB**) aims to provide anomaly detection data sets for physical experiments of water circulation systems. The authors collect data from various sensors, including acceleration, current, pressure,

[11] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[12] https://www.unb.ca/cic/datasets/nsl.html.

[13] https://research.unsw.edu.au/projects/unsw-nb15-dataset.

[14] https://unsworks.unsw.edu.au/entities/dataset/fa64cd6b-4a80-49bb-80fe-23549a60d695/full.

[15] https://secplab.ppgia.pucpr.br/?q=trabid.

[16] https://www.unb.ca/cic/datasets/ids-2017.html.

[17] https://www.unb.ca/cic/datasets/ids-2018.html.

[18] https://sites.google.com/view/iotbotnetdatset/home.

[19] https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/.

[20] https://github.com/waico/SKAB.

temperature, etc. In addition to normal scenarios, they also run the experiments with several anomalous settings, such as partly closed valves that affect the circulation process. Other than the previous data sets where labeling is based on the time of attacks, the data set provides two types of anomaly labels on the granularity of single events, one to mark outliers, i.e., single point anomalies, and another one for change-points, i.e., anomalies that are made up of multiple data instances and indicate changes of long-term trends.

The **Gas Pipeline**[21] data set was collected from a virtual testbed representing a gas pipeline that involves simulations for the physical processes, the network, programmable logic controllers, and human–machine interfaces. Similar to the water treatment systems, the authors collect various sensor values from the simulated cyber-physical systems and design attack scenarios where the components making up the testbed are maliciously modified.

The Gasoil heating loop[22] (**GHL**) data set stems from a simulation where gasoil is heated and transferred between tanks. The authors carry out several runs of normal and expected behavior of the heating loop and consider unauthorized changes of certain settings such as the maximum temperature of pump frequency as anomalous, which they label based on the entire simulation run.

The HIL-based Augmented ICS[23] (**HAI**) security data set comprises data from four different processes: a boiler process, a turbine process, a water treatment process, and a HIL simulation that combines the three other processes. The data is collected from multiple sensors and actuators such as valves. Several attack scenarios are designed by the authors, most of which assume that an attacker manipulates the controller and algorithms that steer the process loop.

Some authors evaluate their approaches with data sets that are not publicly available. For example, Hong et al. [49] describe an avionics testbed that generates categorical data as well as a robot with physical sensors and actuators that generates continuous data.

7.1.4 OSINT and intrusion alerts

The SANS Internet Storm Center provides an openly accessible interface to gather the daily number of attacks recorded by the **DShield**[24] sensor network. The data has been used to detect unexpected spikes in the time-series as anomalies that possibly indicate emerging cyber threats. We point out that this is the only alert data set that is used for the purpose of anomaly detection, and the only one among them that lacks labels. The remaining data sets discussed in this section are primarily used to evaluate approaches for modeling and forecasting.

For example, the **IDEA**[25] data set comprises intrusion alerts collected from an alert sharing platform. The alerts are provided as JSON objects that hold diverse but detailed information on involved IP addresses and ports, nodes, and the type of identified attack. Compared to other alert data sets, this one only spans over one week and is unusually short, but at the same time provides a fine-grained view on the collected alerts since each alert can be analyzed individually.

Data Driven Security[26] (**DDS**) provides an alert data set that is collected from a collection of honeypots that are dispersed across the globe. The data set has been analyzed by counting the number of attacks in daily and hourly time windows. Thereby, it has been found that many alerts appear in very short time intervals, meaning that hourly aggregation is a good choice to capture these extreme attacks.

A vulnerability data set used by several reviewed publications is provided by the National Vulnerability Database[27] (**NVD**), which is maintained by the National Institute of Standard and Technology (NIST). Entries in the data set follow the Common Vulnerabilities and Exposures (CVE) system and provide several references to external resources as well as a human readable description. The data set is JSON formatted with diverse fields; thus, there is no obvious number of dimensions. The popularity of this data set is explained by its completeness and long duration [58].

Hackmageddon[28] provides daily statistics on reported cyber attacks, including some attributes such as the type of attacks. As pointed out in literature [132], the data set is not complete since many attacks are not reported and may even involve incorrect information. Nonetheless, the data set does provide a sample of the threat landscape and is thus useful for certain types of analyses.

Privacy Right Clearinghouse[29] (**PRC**) provides a database of reported data breaches in the United States of America, where many details such as the affected company, its location and industry sector, as well as attack types and breach sizes (i.e., number of affected accounts) are available. The original data set is available online behind a paywall, however, some authors have also provided openly accessible excerpts from that data set.[30]

[21] https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets.

[22] https://kas.pr/ics-research/dataset_ghl_1.

[23] https://github.com/icsdataset/hai.

[24] https://www.dshield.org/data/port.html.

[25] https://data.mendeley.com/datasets/p6tym3fghz/1.

[26] https://datadrivensecurity.info/blog/pages/dds-dataset-collection.html.

[27] https://nvd.nist.gov/.

[28] https://www.hackmageddon.com/category/security/cyber-attacks-statistics/.

[29] https://privacyrights.org/data-breaches.

[30] https://github.com/alan-turing-institute/dynamic_cyber_risk/.

The Open Science Framework[31] (**OSF**) hosts a data set of weekly counts of cyber reports. The data set also includes the average lengths of reports from each week, but according to Bakdash et al. [13], this information does not contribute to fitting time-series models. The original reports have been manually verified at a large Computer Security Service Provider for the U.S. Department of Defense. The authors point out that their data set is of high value since it is unlikely to contain false positives, however, also state that they are unable to provide the original data containing more fine-granular information on the reports due to security reasons.

7.2 Data pre-processing

The data sets discussed in the previous section come in highly different formats, ranging from highly structured numeric data that is straightforward to represent as time-series to unstructured event messages or semi-structured alert objects that require pre-processing to derive data that is suitable for time-series analysis. Figure 4 depicts sample data for these two extremes. The sample in the top of the figure shows simplified log events from the HDFS data set, which comprise a timestamp (note that *081109 203518* is the encoded timestamp for 2008-11-09 20:35:18) and an unstructured event message that corresponds to a certain type of event, e.g., all lines starting with "Receiving block" could belong to the same event type. Parsing the events this way yields sequences, i.e., discrete data points that are temporally and chronologically ordered. At this point, authors generally apply analysis techniques that neglect the exact timestamp and only consider the sequential information conveyed by the event occurrences, such as LSTMs.

An alternative to that strategy is to run time-windows on the parsed event data and to count the number of occurrences for each event type, resulting in a a sequence of so-called event count vectors. Thereby, time-windows can either be moved in fixed time intervals where the step size is the same length as the time-window itself, or in smaller step sizes to achieve a sliding window effect, which means that events can be part of multiple time-windows [46]. Either way, this pre-processing strategy yields regular time-series where the step size determines the granularity of data observations, making it straightforward to analyze the time-series with most techniques, such as statistical approaches. Based on our review, counting is the preferred method to handle all sorts of discrete data, such as alert data sets that involve categorical alert types as well as other discrete features.

When it comes to sensor readings from cyber-physical application areas, data sets often already contain numeric features that form time-series without any need for pre-processing. For example, the sample in the bottom of Fig. 4 is taken from the SKAB data set, which contains a timestamp and several numeric features that yield multi-variate time-series. Note that in this case, the time-series may possibly irregular when measurements are taken in varying intervals or missing, for example, the sample from the SKAB data set lacks the timestamp 2020-02-08 13:30:50. One way to overcome this issue is to apply run time-windows on the data, which is analogous to event counting except that aggregation techniques suitable for numeric data are applied. For example, Kohlrausch et al. [61] suggest averaging of numerical values in time-windows and Li et al. [70] down-sample a data set containing an observation for each second by taking the median of the features in 10 s intervals. Especially for long-term data sets such as observations of cyber incidents or vulnerability occurrences, events may appear in irregular intervals and include lengthy time spans that lack any events. These sparse time-series are often difficult to process even with time-window aggregation strategies (cf. Sect. 5.9.2).

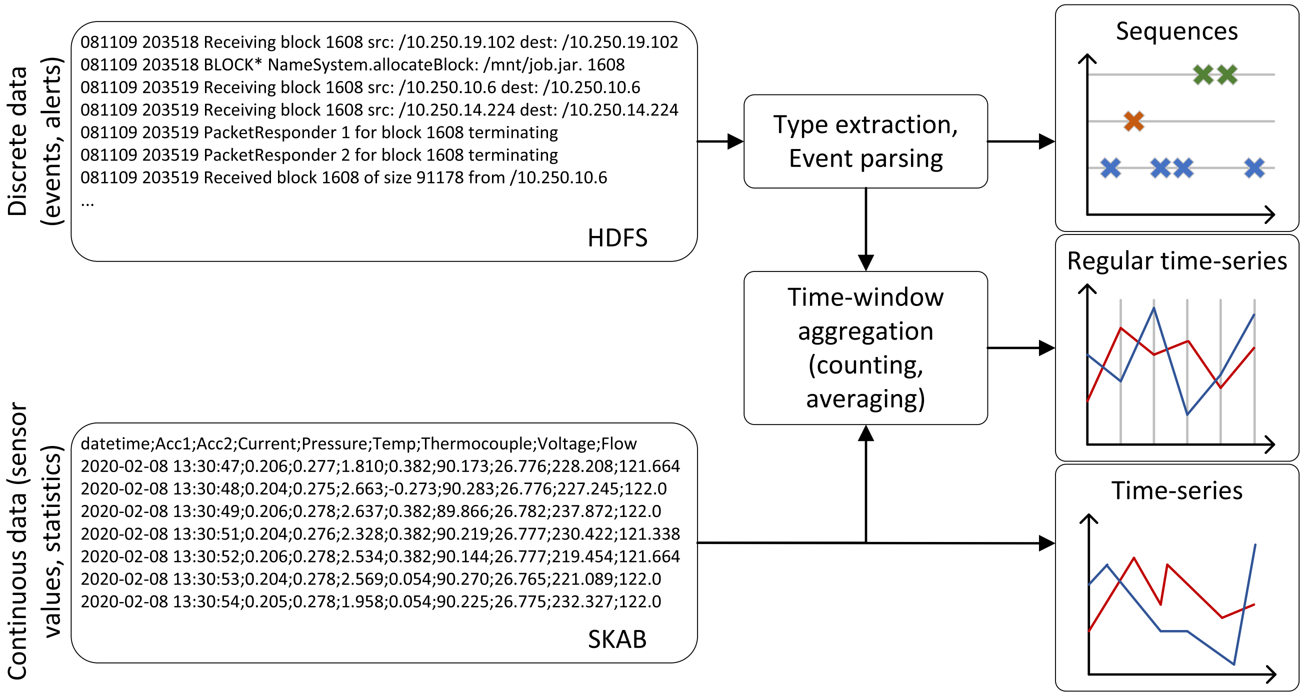
The types of features that are most commonly processed by the approaches proposed in the reviewed literature are linked to the application domain. This is reflected in Fig. 5, which shows the relative frequencies of the various types of features that are considered in each of our identified application domains. As visible in the plot, detective approaches for ICS, VANET, and network intrusions, mostly rely on numeric, binary, categorical, and discrete features that are directly extracted from the analyzed events. On the other hand, host-based intrusion detection as well as failure detection are generally applied with unstructured data and thus more often used with event counts. Analytical approaches that process alert, incident, or intelligence data are most often applied to predict the frequencies of these instances in the future and thus also mostly rely on count data.

Several of these pre-processing strategies are sometimes also used in combination. Meng et al. [81] uses event counts but additionally relies on sequential information of event occurrences for anomaly detection. Other pre-processing strategies that are sometimes applied on the data sets include one-hot encoding to transform categorical features into binary ones [7], normalization of continuous features [7, 38], correction of delayed timestamps [33], dimension reduction [44, 70], removal of outliers [136], and noise reduction [80].

7.3 Evaluation metrics

This section outlines common evaluation metrics used in scientific evaluations. We differentiate between metrics to evaluate the performance of anomaly detection approaches and the prediction performance in analytical approaches.

[31] https://osf.io/hjffm/.


Discrete data (events, alerts)
HDFS
Continuous data (sensor values, statistics)
SKAB
Type extraction, Event parsing
Time-window aggregation (counting, averaging)
Sequences
Regular time-series
Time-series

```
081109 203518 Receiving block 1608 src: /10.250.19.102 dest: /10.250.19.102
081109 203518 BLOCK* NameSystem.allocateBlock: /mnt/job.jar. 1608
081109 203519 Receiving block 1608 src: /10.250.10.6 dest: /10.250.10.6
081109 203519 Receiving block 1608 src: /10.250.14.224 dest: /10.250.14.224
081109 203519 PacketResponder 1 for block 1608 terminating
081109 203519 PacketResponder 2 for block 1608 terminating
081109 203519 Received block 1608 of size 91178 from /10.250.10.6
...
```

```
datetime;Acc1;Acc2;Current;Pressure;Temp;Thermocouple;Voltage;Flow
2020-02-08 13:30:47;0.206;0.277;1.810;0.382;90.173;26.776;228.208;121.664
2020-02-08 13:30:48;0.204;0.275;2.663;-0.273;90.283;26.776;227.245;122.0
2020-02-08 13:30:49;0.206;0.278;2.637;0.382;89.866;26.782;237.872;122.0
2020-02-08 13:30:51;0.204;0.276;2.328;0.382;90.219;26.777;230.422;121.338
2020-02-08 13:30:52;0.206;0.278;2.534;0.382;90.144;26.777;219.454;121.664
2020-02-08 13:30:53;0.204;0.278;2.569;0.054;90.270;26.765;221.089;122.0
2020-02-08 13:30:54;0.205;0.278;1.958;0.054;90.225;26.775;232.327;122.0
```

Fig. 4 Overview of common pre-processing strategies to derive time-series from discrete or continuous input data

7.3.1 Detective approaches

The evaluation of anomaly detection techniques usually focuses on their ability to correctly recognize expected changes of time-series while at the same time maintaining a low false alarm rate. Accordingly, authors count the true positives (TP) as correctly detected anomalous instances, false positives (FP) as incorrectly detected non-anomalous instances, true negatives (TN) as correctly undetected non-anomalous instances, and false negatives (FN) as incorrectly undetected anomalous instances.

Based on these counts, it is then common to compute precision (cf. Eq. 28), recall (cf. Eq. 29), false positive rate (cf. Eq. 30), and false negative rate (cf. Eq. 31). The F1-score (cf. Eq. 32) is a combined metric that considers both precision and recall and accuracy (cf. Eq. 33) as well as classification error (cf. Eq. 34) combine all counts. Since anomaly detection is usually conducted on highly imbalanced data where the normal instances outnumber the anomalous ones, some authors suggest to use Matthews correlation coefficient (cf. Eq. 35) that is more robust against this bias than aforementioned metrics [21].

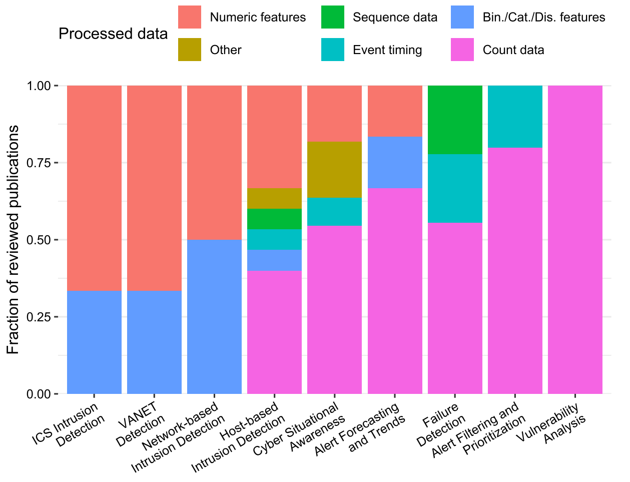


Fig. 5 Relative frequencies of processed data features in application domains

$$P = \frac{TP}{TP + FP} \tag{28}$$

$$R = \frac{TP}{TP + FN} \tag{29}$$

$$FPR = \frac{FP}{FP + TN} \tag{30}$$

$$FNR = \frac{FN}{FN + TP} \tag{31}$$

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{32}$$

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{33}$$

$$CE = \frac{FP + FN}{TP + TN + FP + FN} \tag{34}$$

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{35}$$

Hwang et al. [53] state that when dealing with time-series data, it is necessary to consider both how many different attacks a detector is able to recognize in addition to how well each of these attacks are detected. They therefore suggest two new metrics, time-series aware recall (TaR) that combines the fraction of detected anomalies among all anomalies and the average ratio of detected parts in each anomaly, and time-series aware precision (TaP) that combines the fraction of correct predictions and the average ratio of the correct part at each prediction. We refer to their publication for more details on these metrics.

When detecting anomalies in aggregated data using techniques from time-series analysis, there is sometimes a certain delay until a change of the baseline is reported. Hong et al. [49] therefore define a time window that starts after the attack time in which detections are expected. In addition, they use another time window to aggregate multiple alerts that occur close together. In general, the time-to-detect is another relevant metric for evaluating anomaly detection systems [80]. Other than that, also the time it takes to train and run a model is a metric worth investigating [21].

7.3.2 Analytical approaches

Several metrics that evaluate the quality of time-series forecasts have been proposed in the past [55]. These are suitable to evaluate prediction approaches that aim to forecast the number of vulnerabilities or alerts occurring in a given time interval. Using the predicted value $\hat{y}_t$ and the actual value y_t of any such time-series of length N at time t, the most widely used error metrics in the reviewed literature are the mean absolute error (cf. Eq. 36), root mean square error (cf. Eq. 37), mean absolute percentage error (cf. Eq. 38), and symmetric mean absolute percentage error (cf. Eq. 39) [13, 86, 96, 100, 132, 141].

$$MAE = \frac{1}{N}\sum_{t}^{N}\left|y_t - \hat{y}_t\right| \tag{36}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{t}^{N}\left(y_t - \hat{y}_t\right)^2} \tag{37}$$

$$MAPE = \frac{1}{N}\sum_{t}^{N}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{38}$$

$$SMAPE = \frac{2}{N}\sum_{t}^{N}\left|\frac{y_t - \hat{y}_t}{y_t + \hat{y}_t}\right| \tag{39}$$

As pointed out by some authors, MAE and RMSE are better choices than other metrics when it comes to handling outliers [58] and zero-inflated time-series [135], i.e., time-series where many values are 0, which is frequently the case in security data. A more complex metric that considers multiple periods is the mean absolute scaled error (MASE) [50].

Zängerle et al. [138] point out that standard metrics such as the ones mentioned before are not applicable for probabilistic prediction. They therefore suggest to use the ranked probability score, which is defined as the sum of squared differences between the cumulative forecast probabilities and the observations.

7.4 Reproducibility

In course of reviewing the publications included in this study, we checked whether authors make use of publicly available data sets and ensure access to artifacts such as source code in order to facilitate reproducibility of their work and allow others to validate and extend their approaches or use them for benchmarking. Unfortunately, we found that hardly any publications provide replication packages or links to external resources such as online repositories with instructions on how to reproduce the reported results [50]. In addition, more than 30% of the reviewed publications do not use at least one publicly available data set for their evaluations, meaning that even though the approach could be reimplemented based on the descriptions in the papers, it remains infeasible to validate and assess the presented results.

There are some notable exceptions to this trend, such as the work by Dzhamtyrova et al. [32], who provide both scripts and data that were used for their study. Bakdash et al. [13] state that they cannot publish the raw data for confidentiality reasons, but they do provide a modified version of their data set as well as the source code for partial reproduction of their results.

8 Discussion

In this section we answer our research questions based on the insights from our literature review and state recommendations for future work.

8.1 Answers to research questions

This section answers our research questions RQ1–RQ3.

8.1.1 RQ1: Which domains of cyber security analytics involve time-series analysis?

We discovered through our broad review of scientific literature dealing with time-series analysis in cyber security contexts that there are several relevant application domains, each with their own techniques, goals, and requirements. Table 6 summarizes our findings. We noticed that it is possible to place application domains on a spectrum where fast and automated approaches for the immediate detection of attacks on local data sources are on one end while manually designed long-term forecasts of the overall threat landscape for strategic planning are on the other end.

Detective approaches may be further categorized based on the location where they are deployed, such as hosts, networks, cyber-physical systems, or vehicles. Failure detection that goes beyond the detection of cyber attacks and aims to recognize any anomalous behavior usually analyzes software applications. All of these approaches have in common that time-series analysis techniques are applied for the purpose of anomaly detection, where time-series models representing the previously observed normal behavior are used to check the currently observed system activities using a short time horizon of only seconds to minutes, which aligns with the need to discover attacks and adjust to changes of normal behavior patterns as quickly as possible.

The approaches on the other end of the spectrum do not assume that there are anomalies in the data; instead, they aim to fit a time-series model as closely as possible to time-series that stretch over many months or years. Two main domains are apparent. First, analysis of vulnerabilities that is concerned with the the emergence of potential weaknesses in software applications. Second, cyber situational awareness that considers attack cases and aims to assess and predict their frequencies, damage, and associated risks. In either case, the predicted time horizon is usually in the range of months of even years.

In between the two ends of the spectrum are approaches that analyze alert data, i.e., the output of intrusion detection systems. We differentiate between two domains. The first domain involves approaches that deal with the detection of unusual alerts that are of potential interest to operators. Similar to the detection of anomalies in raw event data, this requires efficient algorithms that usually analyze streams of alert data with a time horizon of some minutes to multiple hours. Rather than identifying anomalies, the second domain is only concerned about the prediction of the number of alerts in the future, which usually involves much longer time horizons of several hours to multiple weeks.

8.1.2 RQ2: What are suitable sources of security-relevant data for time-series analysis?

Table 6 provides an overview of common data sources for each application domain. Detection of failures and host-based intrusions both operate on log data that is generated by or on systems, such as operating system logs, audit logs, syslog, application logs, etc. These are usually unstructured event messages that require parsing or some kind of derivation of event types in order to obtain data in a more structured format. While it is possible to consider the resulting data as chronologically ordered sequences, most approaches then apply time-window based counting in order to derive regular time-series. Thereby, each event type may be regarded as a feature and the obtained time-series is thus suitable for multivariate analyses. Less commonly, numeric values from single events, such as message sizes, or across events, such as event inter-arrival times, are used to derive time-series.

Similarly, alert data contains discrete features such as alert types, IP addresses, host names, etc. Sequential patterns of alert occurrences are usually not considered in approaches that focus on the application of methods from time-series analysis. Instead, the creation of count vectors is the default method to derive time-series from data sources such as alert sharing platforms or collected IDS output.

Network traffic, sensor readings from ICS and cyber-physical systems, and messages sent by vehicles involve structured numeric features that naturally form time-series. However, binary or categorical features such as status codes as well as missing data may still require pre-processing in order to apply methods from time-series analysis that require complete and numerical data.

While data from aforementioned sources are usually collected once from a controlled environment to obtain an evaluation data set, databases of vulnerability and incident reports comprise entries from real threats with global relevance. Again, count vectors are the most popular way to derive time-series from these data sources. However, also additional information such as breach sizes or financial impact are suitable inputs for analysis.

8.1.3 RQ3: What constraints and challenges do data and use-cases impose on the application of time-series analysis and how are they overcome by approaches?

There are several challenges that arise from the application of time-series analysis in cyber security domains. In particular, the nature of the data used to generate time-series appears to make application of common techniques from time-series analysis non-trivial, often requiring analysts to explicitly model certain aspects to achieve adequate fitting and prediction accuracy. Our review shows that there is no clear tendency towards certain analysis techniques for any of the identified application areas; on the contrary, statistical approaches, neural networks, as well as approaches based on conventional machine learning and state-based methods are all used across various domains and data sources. In the following, we summarize the challenges that are encountered by authors in the reviewed publications and propose solutions. We point out that the mentioned solutions are mere

Table 6 Overview of cyber security application domains using time-series analysis techniques

Domain	Method	Time horizon	Data sources
Failure detection	Anomaly detection	Seconds-Minutes	System and application logs
Host-based intrusion detection	Anomaly detection	Seconds-Minutes	System and application logs
Network-based intrusion detection	Anomaly detection	Seconds-Minutes	Network traffic and packet captures
ICS intrusion detection	Anomaly detection	Seconds-Minutes	Cyber-physical sensor readings and actuator statuses
VANET detection	Anomaly detection	Seconds-Minutes	Vehicle sensor readings and communication data
Alert filtering and prioritization	Anomaly detection	Minutes-Hours	IDS alerts and system metrics
Alert forecasting and trends	Prediction	Hours-Weeks	Alert sharing platforms
Cyber situational awareness	Prediction	Hours-Years	Incident reports and open-source intelligence
Vulnerability analysis	Prediction	Months-Years	Vulnerability databases

suggestions derived from the reviewed literature and cannot be regarded as definitive or exhaustive.

- **Data types**. *Challenge:* Collected data sets do not only comprise continuous features, but often involve categorical or discrete features that correspond to event types, status codes, etc. In addition, data points often correspond to event occurrences with irregular-spaced time intervals and may also involve simultaneous occurrences. Unfortunately, many common time-series methods are designed to operate only on time-series with numerical features measured in regular intervals, and are thus unable to handle security data sets [137]. *Solution:* As discussed in Sect. 7.2, categorical values are transformed into continuous time-series by measuring their occurrence counts in time windows [46]. When these windows are moved in regular intervals, this strategy also resolves the issue of dealing with events occurring in irregular time intervals as the resulting time-series is evenly spaced.
- **Time window sizes**. *Challenge:* The aforementioned strategy for the generation of count vectors introduces an additional parameter—the length of the time window—which is critical for prediction and detection accuracy, but often difficult to determine [89, 128]. Specifically, the window should be short enough to capture fine-granular patterns and enable timely detection, but large enough to avoid the generation of spars time-series with many gaps [61]. *Solution:* Most authors address this problem by evaluating with multiple time windows and comparing the results in terms of detection or prediction accuracy [80, 108, 128]. Alternatively, it is possible to use multiple window sizes that are applied in parallel and processed by the model simultaneously [60].
- **Zero-inflated time-series**. *Challenge:* Zero-inflated time-series, sometimes also referred to as sparse time-series, have an overabundance of zeros and exhibit high volatility when values occur. They are common in security data and may be generated when time window sizes used for event counting are large enough so that no events occur within that interval [50, 56]. Many models are not designed to process such time-series, for example, ARIMA models rely on a Gaussian noise process, which is commonly violated by zero-inflated time-series [61]. Some evaluation metrics are also undefined in case that no observations occur within a time window [135]. *Solution:* Use models that are known to adequately handle zero-inflated time-series, such as Croston's method (cf. Sect. 6.1.6), neural networks (cf. Sect. 6.3) [135], GARCH models [97], or extensions to well-known methods such as autoregressive models [82]. For evaluation, rely on metrics that are well-defined for empty time windows, such as the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) (cf. Sect. 7.3.2).
- **Imbalanced data sets**. *Challenge:* Event occurrences are often highly imbalanced, meaning that some event types occur more frequently than others. Similarly, the number of anomalous data instances is usually far smaller than the set of normal or benign instances in the data sets [142]. *Solution:* Specifically during evaluation, it is vital to make use of metrics that are known to be robust against imbalanced classes and suitable for evaluation of tasks such as anomaly detection (cf. Sect. 7.3.1). In addition, resampling of the data set is a common way in machine learning tasks to obtain more balanced data sets [35], and some resampling strategies that have been specifically designed for time-series data could be useful for security data [83].
- **Multiple features**. *Challenge:* Multivariate data sets often involve more than one feature with inter-correlation between channels [60] or non-linear relationships [70]. While it is always possible to resort to univariate analysis that considers each feature separately [16], incorporat-

ing these dependencies could be important for prediction and detection accuracy and should thus not be ignored [9]. *Solution:* Resort to time-series analysis techniques that support multivariate data, such as Recurrent Neural Networks (RNN; cf. Sect. 6.3.2) or Generative Adversarial Networks (GAN; cf. Sect. 6.3.3). Other suitable techniques suggested by state-of-the-art surveys include Variational AutoEncoders, Hidden Markov Models, or similarity-based models [16].

- **Seasonality**. *Challenge:* Event occurrences are often linked to scheduled or human activities that have periodically recurring behavior [60]. In fact, data may involve multi-seasonality, i.e., periodic behavior with several interval times that act simultaneously on the data such as monthly and yearly patterns [133]. Moreover, different channels or event types may also be affected by different seasonality in the same data set [142] and correlations within the data may involve short-term to long-term dependencies [132]. *Solution:* Utilization of models that integrate seasonality or explicitly model periodic patterns (cf. Sect. 6.1.5), such as Triple Exponential Smoothing (TES) (cf. Sect. 6.1.1) or neural networks (cf. Sects. 6.3.2 and 6.3.3). Other techniques for modeling of seasonal patterns that we did not encounter in the reviewed literature include seasonal ARIMA (SARIMA) models [28], the Prophet forecasting model that integrates multi-seasonality through Fourier analysis [118], as well as decompositions of time-series with additive or multiplicative seasonality components [28].
- **Bias**. *Challenge:* Delays between occurrences of events, such as cyber incidents, and their appearance in data sets, such as attack reports, introduce a bias that may deteriorate prediction or detection performance. *Solution:* This situation can be alleviated by estimating the delay based on past examples and incorporating it in the model [33].
- **Data contamination**. *Challenge:* The generation of real-world data can hardly be controlled. Accordingly, the collected data is often affected by contamination, i.e., anomalous patterns that randomly occur in data that is supposed to comprise only normal or benign behavior. Training machine learning models with such data may deteriorate the prediction or detection performance. *Solution:* Neural networks have been used to counteract the influence of contamination in the data [60, 133]. Moreover, signal processing provides several techniques for noise removal in time-series, including filters and wavelet thresholding [34].
- **Missing data**. *Challenge:* Data sets may be incomplete when errors occur during the collection process or sensors fail to produce or transfer measured values. *Solution:* As discussed in Sect. 6.3.3, GANs have been shown to be able to reconstruct data from the remaining data set [137]. Several other techniques to handle missing data have been investigated in literature, including Kalman Filters, expectation-maximization, dynamic time warping, SARIMA, RNNs, and many more [4].
- **Lack of labeled data**. *Challenge:* Supervised approaches require labeled data for training. Semi-supervised approaches only require anomaly-free data for training, but labeled data is still necessary to evaluate semi-supervised or unsupervised approaches [70, 133]. *Solution:* Some authors consider pseudo labeling, i.e., labeling using a machine learning model, to obtain labeled data for their evaluation [10, 80].
- **Explainability of results**. *Challenge:* Some machine learning models suffer from low explainability, meaning that it is non-trivial or even infeasible to comprehend the factors that influence the results of predictions or detections [49]. Unfortunately, poor explainability of models or results may lead to reluctance of using or relying on the methods [50]. *Solution:* Explainability is an actively researched field in machine learning; recently, some methods to improve explainability even for complex classifiers such as neural networks have been proposed [14, 75].
- **Diverse anomaly types**. *Challenge:* Anomalies in time-series may take many forms [49]. For example, anomalies could be single data points that are outliers such as bursts [44], changes of periodically occurring behavior, collections of data points that are only considered anomalous as groups but not individually [104], changes of long-term trends [67], as well as changes in sequential patterns of event types [81]. *Solution:* Some detection techniques are designed for generic types of anomalies, such as neural networks that recognize all sorts of unusual patterns (cf. Sect. 6.3). Alternatively, reported anomalies of several detection methods, each focusing on a specific type of anomaly, may be combined.
- **Scalable and automatic algorithms**. *Challenge:* Analyzed data sets are way too large and versatile to be analyzed manually and require solutions that operate in an automatic manner. Moreover, data should be monitored continuously as real-world applications are often expected to run indefinitely and enable real-time detection [44, 60, 67, 80, 86, 133]. *Solution:* Ensure that developed algorithms are efficient in processing data points and support online analysis, i.e., processing only takes a single pass over the data such as incremental algorithms [16].

8.2 Future directions

The challenges of applying time-series in security applications stated in our answer to RQ3 are only partially resolved by the solutions proposed in the reviewed literature and thus leave several research opportunities for future work. For example, running time-windows over the data to generate event counts is difficult when groups of related events and seasonal patterns have high variances in terms of duration, because relevant time intervals could either only make up a small fraction of the entire-time window or be split in chunks by the edges of time-windows. Rather than only relying on count data, authors could thus consider additional methods to identify anomalies, in particular, using techniques from change point detection [6, 121].

Another interesting analysis method that is capable of processing event data that occurs in irregular intervals is Bayesian Binning [103], which aims to find optimal segmentation of the time-series with respect to local variability. Advantages of this method include the fact that it does not rely on predefined time window sizes and its robustness to noise, e.g., normal background behavior that makes it difficult to identify anomalies.

During our review, we noticed that approaches using neural networks suffer from low explainability in comparison to conventional methods. While models such as ARIMA enable to compute prediction intervals based on simple statistical measures, the output of an LSTM RNN is not as trivial to understand given some complex input values. It could therefore be beneficial if authors that apply neural networks on time-series also employ methods for Explainable Artificial Intelligence (XAI) that have been specifically developed for that kind of data [99]. Other challenges are also not sufficiently addressed by the current state-of-the-art. In particular, we suggest to search for time-series models applied in other research fields that could resolve prevalent issues with properties of security time-series, such as patterns corresponding to multiple seasonal periods, data sets that contain missing events, or the influence of noise on the capabilities of models to fit relevant data patterns.

Our review of common data sets in Sect. 7.1 shows that there exist multiple publicly available data sets from various security domains that are suitable for time-series analysis. Data sets that are used to evaluate anomaly-based approaches usually contain traces of several attack cases, which are subject of detection. To the best of our knowledge, there is no structured overview that analyses the various manifestations of different types of attacks in the monitored data. Accordingly, it is difficult to estimate which attacks are the most relevant when it comes to detection with time-series models, such as attacks that produce high numbers of events that could be recognized through event counts or attacks that affect numeric event parameters that are suitable to extract time-series. We therefore propose to create such an overview of attack manifestations that considers diverse attack strategies and log sources to better understand what types of detection strategies can be effectively deployed against certain types of attacks.

Recently, researchers have started to investigate adversarial attacks against various types of machine learning models for time-series analysis. Thereby, time-series are modified with small perturbations to influence the classification or detection output of a model [36, 59, 98]. In general, more complex models such as deep neural networks are considered to be more susceptible to adversarial attacks than models based on conventional machine learning or statistics [88]. Given the rising trend of employing neural network models in time series analysis in combination with the growing complexity of model architectures, adversarial attacks pose an increasingly significant threat. Some defense strategies include training on adversarial samples that are included in the training set [78], randomization of training samples that is achieved by adding noise [25], and projecting the input to the neural network on a lower dimensional vector to filter out the noise added during the generation of adversarial samples [57]. In addition to further research on the generation and mitigation of adversarial samples, we consider it interesting to investigate how perturbations of time-series can take place in data sets comprising security event data in real-world settings, e.g., through the injection of certain event types.

9 Conclusion

Time-series analysis provides a versatile set of techniques, including statistical methods, neural networks, conventional machine learning, and state-based models, which are highly useful for many applications across diverse cyber security domains. In course of the literature review presented in this paper, we identified the following relevant security application areas and suitable data sources: (i) failure detection in system log data, (ii) host-based intrusion detection in system log data, (iii) network-based intrusion detection in network traffic, (iv) ICS intrusion detection in cyber-physical data, (v) VANET detection in vehicle sensor data, (vi) alert filtering and prioritization in IDS alert data sets, (vii) alert forecasting and trends in shared alert databases, (viii) cyber situational awareness on incident data sets, and (ix) analysis of software vulnerabilities. Our review suggests that most scientific approaches leveraging TSA fall in one of two categories: detective approaches that aim to automatically and timely disclose anomalies in event or alert data and predictive approaches that rely on manual modeling of long-term trends to forecast future occurrences of alerts, attacks, or vulnerabilities. Several properties of security data sets emerged from our review as challenges for time-series analysis, including mixtures of numerical and categorical features, sparse time-series with high volatility, high-dimensional data, and seasonality. Our analysis suggests that it is vital that analysts consider these issues, for example, by applying models that are particularly designed to adequately handle certain data properties or transforming discrete data into numeric count vectors. For future research, we recommend to analyze which attack types are the most relevant ones for anomaly detection with time-series models, work on new pre-processing strategies that do not rely on time-windows but are more generally applicable, and propose methods that address specific properties of security time-series such as irregularity, sparsity, multi-seasonality, noise, and incompleteness.

Author Contributions M.L. designed the study, conducted the literature review, and wrote the main manuscript text. F.S. contributed to the sections on Cyber Situational Awareness and Vulnerability Analysis. B.S. added the sections on Network-based Intrusion Detection. A.F. reviewed literature on adversarial samples and added the respective paragraphs. T.U. contributed to the section on statistical models for time-series analysis. All authors contributed to the background section. Finally, all authors reviewed the manuscript.

Funding Open access funding provided by AIT Austrian Institute of Technology GmbH. The work in this paper has received funding from the Austrian Research Promotion Agency (FFG) under GA No. FO999899544 (PRESENT).

Data availability No datasets were generated or analysed during the current study.

Declarations

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

Human and animal rights This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Acar, A., Lu, L., Uluagac, A.S., Kirda, E.: An analysis of malware trends in enterprise networks. In: Information Security: 22nd International Conference, ISC 2019, New York City, NY, USA, September 16–18, 2019, Proceedings 22, pp. 360–380. Springer (2019)
2. Ahmad, Z., Shahid Khan, A., Nisar, K., Haider, I., Hassan, R., Haque, M.R., Tarmizi, S., Rodrigues, J.J.: Anomaly detection using deep neural network for iot architecture. Appl. Sci. **11**(15), 7050 (2021)
3. Ahmed, C.M., Palleti, V.R., Mathur, A.P.: Wadi: a water distribution testbed for research in the design of secure cyber physical systems. In: Proceedings of the 3rd International Workshop on Cyber-physical Systems for Smart Water Networks, pp. 25–28 (2017)
4. Ahn, H., Sun, K., Kim, K.P., et al.: Comparison of missing data imputation methods in time series forecasting. Comput. Mater. Continua **70**(1), 767–779 (2022)
5. Alahmadi, B.A., Axon, L., Martinovic, I.: 99% false positives: a qualitative study of {SOC} analysts' perspectives on security alarms. In: 31st USENIX Security Symposium (USENIX Security 22), pp. 2783–2800 (2022)
6. Aminikhanghahi, S., Cook, D.J.: A survey of methods for time series change point detection. Knowl. Inf. Syst. **51**(2), 339–367 (2017)
7. Anastasiadis, D., Lenart, J.: Detection of software incidents from large log material with the use of unsupervised machine learning (2022)
8. Ansari, M.S., Bartoš, V., Lee, B.: Gru-based deep learning approach for network intrusion alert prediction. Futur. Gener. Comput. Syst. **128**, 235–247 (2022)
9. Ara, A., Louzada, F., Diniz, C.A.: Statistical monitoring of a web server for error rates: a bivariate time-series copula-based modeling approach. J. Appl. Stat. **44**(13), 2287–2300 (2017)
10. Arazo, E., Ortego, D., Albert, P., O'Connor, N.E., McGuinness, K.: Pseudo-labeling and confirmation bias in deep semi-supervised learning. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
11. Baddar, S.W.A.H., Merlo, A., Migliardi, M.: Anomaly detection in computer networks: a state-of-the-art review. JoWUA **5**(4), 29–64 (2014)
12. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271 (2018)
13. Bakdash, J.Z., Hutchinson, S., Zaroukian, E.G., Marusich, L.R., Thirumuruganathan, S., Sample, C., Hoffman, B., Das, G.: Malware in the future? forecasting of analyst detection of cyber events. J. Cybersecur. **4**(1), tyy007 (2018)
14. Belle, V., Papantonis, I.: Principles and practice of explainable machine learning. Front. Big Data **4**, 688–969 (2021)
15. Bichara, D., Iggidr, A., Sallet, G.: Global analysis of multi-strains SIS, SIR and MSIR epidemic models. J. Appl. Math. Comput. **44**, 273–292 (2014)
16. Blázquez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM Comput. Surv. (CSUR) **54**(3), 1–33 (2021)
17. Box, G.E.P., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control (Wiley Series in Probability and Statistics). Wiley (2015)
18. Braei, M., Wagner, S.: Anomaly detection in univariate time-series: a survey on the state-of-the-art. arXiv preprint arXiv:2004.00433 (2020)
19. Brockwell, P.J., Davis, R.A.: Time Series: Theory and Methods. Springer, Berlin (1991)
20. Čeponis, D., Goranin, N.: Towards a robust method of dataset generation of malicious activity for anomaly-based hids training and presentation of awsctd dataset. Baltic J. Modern Comput. **6**(3), 217–234 (2018)
21. Čeponis, D., Goranin, N.: Investigation of dual-flow deep learning models lstm-fcn and gru-fcn efficiency against single-flow cnn models for the host-based intrusion and malware detection task on univariate times series data. Appl. Sci. **10**(7), 2373 (2020)
22. Chalapathy, R., Chawla, S.: Deep learning for anomaly detection: a survey. arXiv preprint arXiv:1901.03407 (2019)
23. Chang, Y.Y., Zavarsky, P., Ruhl, R., Lindskog, D.: Trend analysis of the cve for software vulnerability management. In: 2011 IEEE 3rd International Conference on Privacy, Security, Risk and Trust

and 2011 IEEE 3rd International Conference on Social Computing, pp. 1290–1293. IEEE (2011)
24. Choi, K., Yi, J., Park, C., Yoon, S.: Deep learning for anomaly detection in time-series data: review, analysis, and guidelines. IEEE Access **9**, 120043–120065 (2021)
25. Cohen, J.M., Rosenfeld, E., Kolter, J.Z.: Certified adversarial robustness via randomized smoothing (2019)
26. Condon, E., He, A., Cukier, M.: Analysis of computer security incident data using time series models. In: 2008 19th International Symposium on Software Reliability Engineering (ISSRE), pp. 77–86. IEEE (2008)
27. Cook, A.A., Mısırlı, G., Fan, Z.: Anomaly detection for iot time-series data: a survey. IEEE Internet Things J. **7**(7), 6481–6494 (2019)
28. Dama, F., Sinoquet, C.: Time series analysis and modeling to forecast: a survey. arXiv preprint arXiv:2104.00164 (2021)
29. Dhanabal, L., Shantharajah, S.: A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. Int. J. Adv. Res. Comput. Commun. Eng. **4**(6), 446–452 (2015)
30. Dodiya, B., Singh, U.K., Gupta, V.: Trend analysis of the cve classes across cvss metrics. Int. J. Comput. Appl. **975**, 8887 (2021)
31. Du, S., Cao, J.: Behavioral anomaly detection approach based on log monitoring. In: 2015 International Conference on Behavioral, Economic and Socio-cultural Computing (BESC), pp. 188–194. IEEE (2015)
32. Dzhamtyrova, R., Maple, C.: Dynamic cyber risk estimation with competitive quantile autoregression. Data Min. Knowl. Disc. **36**(2), 513–536 (2022)
33. Eling, M., Ibragimov, R., Ning, D.: Time dynamics of cyber risk. Available at SSRN 4497621 (2023)
34. Esling, P., Agon, C.: Time-series data mining. ACM Comput. Surv. (CSUR) **45**(1), 1–34 (2012)
35. Estabrooks, A., Jo, T., Japkowicz, N.: A multiple resampling method for learning from imbalanced data sets. Comput. Intell. **20**(1), 18–36 (2004)
36. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Adversarial attacks on deep neural networks for time series classification. In: 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2019)
37. Fernandes, G., Rodrigues, J.J., Carvalho, L.F., Al-Muhtadi, J.F., Proença, M.L.: A comprehensive survey on network anomaly detection. Telecommun. Syst. **70**, 447–489 (2019)
38. Filonov, P., Lavrentyev, A., Vorontsov, A.: Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model. arXiv preprint arXiv:1612.06676 (2016)
39. Finder, I., Sheetrit, E., Nissim, N.: Time-interval temporal patterns can beat and explain the malware. Knowl. Based Syst. **241**, 108–266 (2022)
40. Friedlander, B., Porat, B.: The modified Yule–Walker method of ARMA spectral estimation. IEEE Trans. Aerosp. Electron. Syst. **20**, 158–173 (1984)
41. Fulcher, B.D.: Feature-based time-series analysis. In: Feature Engineering for Machine Learning and Data Analytics, pp. 87–116. CRC press (2018)
42. Fulcher, B.D.: Feature-based time-series analysis. In: Feature Engineering for Machine Learning and Data Analytics, pp. 87–116. CRC press (2018)
43. Goh, J., Adepu, S., Junejo, K.N., Mathur, A.: A dataset to support research in the design of secure water treatment systems. In: Critical Information Infrastructures Security: 11th International Conference, CRITIS 2016, Paris, France, October 10–12, 2016, Revised Selected Papers 11, pp. 88–99. Springer (2017)
44. Granlund, O.: Unsupervised anomaly detection on log-based time series data (2019)
45. Haider, W., Hu, J., Slay, J., Turnbull, B.P., Xie, Y.: Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. J. Netw. Comput. Appl. **87**, 185–192 (2017)
46. He, S., Zhu, J., He, P., Lyu, M.R.: Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE), pp. 207–218. IEEE (2016)
47. Hewamalage, H., Bergmeir, C., Bandara, K.: Global models for time series forecasting: a simulation study. Pattern Recognit. **124**, 108441 (2022)
48. Hipel, K.W., McLeod, A.I. (eds.): Time Series Modelling of Water Resources and Environmental Systems. Elsevier, Amsterdam (1994)
49. Hong, A.E., Malinovsky, P.P., Damodaran, S.K.: Towards attack detection in multimodal cyber-physical systems with sticky hdp-hmm based time series analysis. Digital Threats: Research and Practice (2022)
50. Husák, M., Bartoš, V., Sokol, P., Gajdoš, A.: Predictive methods in cyber defense: current experience and research challenges. Futur. Gener. Comput. Syst. **115**, 517–530 (2021)
51. Husák, M., Komárková, J., Bou-Harb, E., Čeleda, P.: Survey of attack projection, prediction, and forecasting in cyber security. IEEE Commun. Surv. Tutor. **21**(1), 640–660 (2018)
52. Husák, M., Žádník, M., Bartoš, V., Sokol, P.: Dataset of intrusion detection alerts from a sharing platform. Data Brief **33**, 106530 (2020)
53. Hwang, W.S., Yun, J.H., Kim, J., Kim, H.C.: Time-series aware precision and recall for anomaly detection: considering variety of detection result and addressing ambiguous labeling. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, pp. 2241–2244 (2019)
54. Hyndman, R.J., Athanasopoulos, G.: Forecasting: principles and practice. OTexts (2018)
55. Hyndman, R.J., Koehler, A.B.: Another look at measures of forecast accuracy. Int. J. Forecast. **22**(4), 679–688 (2006)
56. Jain, S., Singh, I., Chandra, A., Zhang, Z.L., Bronevetsky, G.: Extracting the textual and temporal structure of supercomputing logs. In: 2009 International Conference on High Performance Computing (HiPC), pp. 254–263. IEEE (2009)
57. Jalal, A., Ilyas, A., Daskalakis, C., Dimakis, A.G.: The robust manifold defense: adversarial training using generative models (2019)
58. Kalouptsoglou, I., Tsoukalas, D., Siavvas, M., Kehagias, D., Chatzigeorgiou, A., Ampatzoglou, A.: Time series forecasting of software vulnerabilities using statistical and deep learning models. Electronics **11**(18), 2820 (2022)
59. Karim, F., Majumdar, S., Darabi, H.: Adversarial attacks on time series. IEEE Trans. Pattern Anal. Mach. Intell. **43**(10), 3309–3320 (2020)
60. Khoshnevisan, F., Fan, Z., Carvalho, V.R.: Improving robustness on seasonality-heavy multivariate time series anomaly detection. arXiv preprint arXiv:2007.14254 (2020)
61. Kohlrausch, J., Brin, E.A.: Arima supplemented security metrics for quality assurance and situational awareness. Digital Threats Res. Pract. **1**(1), 1–21 (2020)
62. Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. Futur. Gener. Comput. Syst. **100**, 779–796 (2019)
63. Kurtz, G.: Crowdstrike 2024 global threat report. Crowdstrike.com **2024**, 1–61 (2024)
64. Kuruvila, A.P., Karmakar, S., Basu, K.: Time series-based malware detection using hardware performance counters. In: 2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 102–112. IEEE (2021)

65. Landauer, M., Onder, S., Skopik, F., Wurzenberger, M.: Deep learning for anomaly detection in log data: a survey. Mach. Learn. Appl. **12**, 100470 (2023)
66. Landauer, M., Skopik, F., Wurzenberger, M., Rauber, A.: System log clustering approaches for cyber security applications: a survey. Comput. Secur. **92**, 101739 (2020)
67. Landauer, M., Wurzenberger, M., Skopik, F., Settanni, G., Filzmoser, P.: Dynamic log file analysis: an unsupervised cluster evolution approach for anomaly detection. Comput. Secur. **79**, 94–116 (2018)
68. Lande, D., Feher, A.: Osint time series forecasting methods analysis. Theor. Appl. Cybersecuri. **5**(1) (2023)
69. Lella, I., Tsekmezoglou, E., Theocharidou, M., Magonara, E., Malatras, A., Naydenov, R.S., Ciobanu, C.: Enisa threat landscape 2023. Eur. Union Agency Cybersecur. **2023**, 1–161 (2023)
70. Li, D., Chen, D., Goh, J., Ng, S.k.: Anomaly detection with generative adversarial networks for multivariate time series. arXiv preprint arXiv:1809.04758 (2018)
71. Li, D., Chen, D., Jin, B., Shi, L., Goh, J., Ng, S.K.: Mad-gan: multivariate anomaly detection for time series data with generative adversarial networks. In: International Conference on Artificial Neural Networks, pp. 703–716. Springer (2019)
72. Li, G., Jung, J.J.: Deep learning for anomaly detection in multivariate time series: approaches, applications, and challenges. Inf. Fusion (2022)
73. Li, X., Chen, P., Jing, L., He, Z., Yu, G.: Swisslog: robust and unified deep learning based log anomaly detection for diverse faults. In: 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), pp. 92–103. IEEE (2020)
74. Li, Z., Xiang, Z., Gong, W., Wang, H.: Unified model for collective and point anomaly detection using stacked temporal convolution networks. Appl. Intell. **52**(3), 3118–3131 (2022)
75. Linardatos, P., Papastefanopoulos, V., Kotsiantis, S.: Explainable AI: a review of machine learning interpretability methods. Entropy **23**(1), 18 (2020)
76. Longobardi, A., Villani, P.: Trend analysis of annual and seasonal rainfall time series in the mediterranean area. Int. J. Climatol. **30**(10), 1538–1546 (2010)
77. Lopes, I.O., Zou, D., Abdulqadder, I.H., Akbar, S., Li, Z., Ruambo, F., Pereira, W.: Network intrusion detection based on the temporal convolutional model. Comput. Secur. **135**, 103465 (2023)
78. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks (2019)
79. Mahdavisharif, M., Jamali, S., Fotohi, R.: Big data-aware intrusion detection system in communication networks: a deep learning approach. J. Grid Comput. **19**, 1–28 (2021)
80. Mahmoud, H., Wu, W., Gaber, M.M.: A time-series self-supervised learning approach to detection of cyber-physical attacks in water distribution systems. Energies **15**(3), 914 (2022)
81. Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., Chen, Y., Zhang, R., Tao, S., Sun, P., et al.: Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: IJCAI, vol. 19, pp. 4739–4745 (2019)
82. Möller, T.A., Weiß, H.C., Kim, H.Y., Sirchenko, A.: Modeling zero inflation in count data time series with bounded support. Methodol. Comput. Appl. Probab. **20**, 589–609 (2018)
83. Moniz, N., Branco, P., Torgo, L.: Resampling strategies for imbalanced time series forecasting. Int. J. Data Sci. Anal. **3**, 161–181 (2017)
84. Morris, T.H., Thornton, Z., Turnipseed, I.: Industrial control system simulation and data logging for intrusion detection system research. In: 7th Annual Southeastern Cyber Security Summit, pp. 3–4 (2015)
85. Moustafa, N., Slay, J.: Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6. IEEE (2015)
86. Naveiro, R., Rodríguez, S., Rios Insua, D.: Large-scale automated forecasting for network safety and security monitoring. Appl. Stoch. Model. Bus. Ind. **35**(3), 431–447 (2019)
87. Nerlove, M., Grether, D.M., Carvalho, J.L.: Analysis of Economic Time Series: A Synthesis. Academic Press (2014)
88. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: high confidence predictions for unrecognizable images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
89. Ohana, D., Wassermann, B., Dupuis, N., Kolodner, E., Raichstein, E., Malka, M.: Hybrid anomaly detection and prioritization for network logs at cloud scale. In: Proceedings of the 17th European Conference on Computer Systems, pp. 236–250 (2022)
90. Okutan, A., Werner, G., McConky, K., Yang, S.J.: Poster: cyber attack prediction of threats from unconventional resources (capture). In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 2563–2565 (2017)
91. Oliner, A., Stearley, J.: What supercomputers say: a study of five system logs. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), pp. 575–584. IEEE (2007)
92. Ott, R.L., Longnecker, M.T.: An introduction to statistical methods and data analysis. In: Cengage Learning (2015)
93. Panchelyuga, V.A., Panchelyuga, M.S., Seraya, O.Y.: On external influences on the radioactive decay rate fluctuations. Metaphysics **4**, 10–34 (2020)
94. Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. ACM Comput. Surv. (CSUR) **54**(2), 1–38 (2021)
95. Patri, O., Wojnowicz, M., Wolff, M.: Discovering malware with time series shapelets. In: Proceedings of the 50th Hawaii International Conference on System Sciences (2017)
96. Pokhrel, N.R., Rodrigo, H., Tsokos, C.P., et al.: Cybersecurity: time series predictive modeling of vulnerabilities of desktop operating system using linear and non-linear approach. J. Inf. Secur. **8**(04), 362 (2017)
97. Qian, L., Zhu, F.: A flexible model for time series of counts with overdispersion or underdispersion, zero-inflation and heavy-tailedness. Commun. Math. Stat. 1–24 (2023)
98. Rathore, P., Basak, A., Nistala, S.H., Runkana, V.: Untargeted, targeted and universal adversarial attacks and defenses on time series. In: 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2020)
99. Rojat, T., Puget, R., Filliat, D., Del Ser, J., Gelin, R., Díaz-Rodríguez, N.: Explainable artificial intelligence (xai) on time-series data: a survey. arXiv preprint arXiv:2104.00950 (2021)
100. Samia, N.K.: Global cyber attack forecast using AI techniques (2023)
101. Sarker, I.H.: Deep cybersecurity: a comprehensive overview from neural network and deep learning perspective. SN Comput. Sci. **2**(3), 154 (2021)
102. Sarker, I.H.: Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. SN Comput. Sci. **2**(6), 420 (2021)
103. Scargle, J.D., Norris, J.P., Jackson, B., Chiang, J.: Studies in astronomical time series analysis. vi. Bayesian block representations. Astrophys. J. **764**(2), 167 (2013)
104. Seong, C., Song, Y., Hyun, J., Cheong, Y.G.: Towards building intrusion detection systems for multivariate time-series data. In: Silicon Valley Cybersecurity Conference, pp. 45–56. Springer (2021)

105. Sgueglia, A., Di Sorbo, A., Visaggio, C.A., Canfora, G.: A systematic literature review of iot time series anomaly detection solutions. Futur. Gener. Comput. Syst. **134**, 170–186 (2022)
106. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP, pp. 108–116 (2018)
107. Shirani, P., Azgomi, M.A., Alrabaee, S.: A method for intrusion detection in web services based on time series. In: 2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 836–841. IEEE (2015)
108. Silva, A., Pontes, E., Zhou, F., Guelf, A., Kofuji, S.: Prbs/ewma based model for predicting burst attacks (brute froce, dos) in computer networks. In: 9th International Conference on Digital Information Management (ICDIM 2014), pp. 194–200. IEEE (2014)
109. Sims, C.A.: Continuous and discrete time models. In: Macroeconometrics and Time Series Analysis, pp. 60–67. Springer (2010)
110. Skopik, F., Wurzenberger, M., Landauer, M.: Smart Log Data Analytics. Springer, Berlin (2021)
111. Spiliotis, E.: Time Series Forecasting with Statistical, Machine Learning, and Deep Learning Methods: Past, Present, and Future, pp. 49–75 (2023). https://doi.org/10.1007/978-3-031-35879-1_3
112. Stefansson, H., Sigmarsdottir, S., Jensson, P., Shah, N.: Discrete and continuous time representations and mathematical models for large production scheduling problems: a case study from the pharmaceutical industry. Eur. J. Oper. Res. **215**(2), 383–392 (2011)
113. Stojanović, B., Božić, J., Hofer-Schmitz, K., Nahrgang, K., Weber, A., Badii, A., Sundaram, M., Jordan, E., Runevic, J.: Follow the trail: machine learning for fraud detection in fintech applications. Sensors **21**(5), 1594 (2021)
114. Stojanović, B., Neuschmied, H., Winter, M., Kleb, U.: Enhanced anomaly detection for cyber-attack detection in smart water distribution systems. In: Proceedings of the 17th International Conference on Availability, Reliability and Security, pp. 1–7 (2022)
115. Suda, H., Natsui, M., Hanyu, T.: Systematic intrusion detection technique for an in-vehicle network based on time-series feature extraction. In: 2018 IEEE 48th International Symposium on Multiple-Valued Logic (ISMVL), pp. 56–61. IEEE (2018)
116. Taormina, R., Galelli, S., Tippenhauer, N.O., Salomons, E., Ostfeld, A., Eliades, D.G., Aghashahi, M., Sundararajan, R., Pourahmadi, M., Banks, M.K., et al.: Battle of the attack detection algorithms: disclosing cyber attacks on water distribution networks. J. Water Resour. Plan. Manag. **144**(8), 04018048 (2018)
117. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6. IEEE (2009)
118. Taylor, S.J., Letham, B.: Forecasting at scale. Am. Stat. **72**(1), 37–45 (2018)
119. Tiwari, T., Turk, A., Oprea, A., Olcoz, K., Coskun, A.K.: User-profile-based analytics for detecting cloud security breaches. In: 2017 IEEE International Conference on Big Data (Big Data), pp. 4529–4535. IEEE (2017)
120. Torres, J.F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., Troncoso, A.: Deep learning for time series forecasting: a survey. Big Data **9**(1), 3–21 (2021)
121. Truong, C., Oudre, L., Vayatis, N.: Selective review of offline change point detection methods. Signal Process. **167**, 107299 (2020)
122. Ullah, F., Babar, M.A.: Architectural tactics for big data cybersecurity analytics systems: a review. J. Syst. Softw. **151**, 81–118 (2019)
123. Ullah, I., Mahmoud, Q.H.: A technique for generating a botnet dataset for anomalous activity detection in iot networks. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 134–140. IEEE (2020)
124. Ullrich, T.: On the autoregressive time series model using real and complex analysis. Forecasting **3**, 716–728 (2022)
125. Verma, R.M., Marchette, D.J.: Cybersecurity Analytics. CRC Press, Boca Raton (2019)
126. Viegas, E.K., Santin, A.O., Oliveira, L.S.: Toward a reliable anomaly-based intrusion detection in real-world environments. Comput. Netw. **127**, 200–216 (2017)
127. Viinikka, J., Debar, H.: Monitoring ids background noise using ewma control charts and alert information. In: Recent Advances in Intrusion Detection: 7th International Symposium, RAID 2004, Sophia Antipolis, France, September 15-17, 2004. Proceedings 7, pp. 166–187. Springer (2004)
128. Viinikka, J., Debar, H., Mé, L., Lehikoinen, A., Tarvainen, M.: Processing intrusion detection alert aggregates with time series modeling. Inf. Fusion **10**(4), 312–324 (2009)
129. Viinikka, J., Debar, H., Mé, L., Séguier, R.: Time series modeling for ids alert management. In: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security, pp. 102–113 (2006)
130. Wang, F., Yang, S., Wang, C., Li, Q.: A novel intrusion detection system for malware based on time-series meta-learning. In: International Conference on Machine Learning for Cyber Security, pp. 50–64. Springer (2020)
131. Wang, X., Smith, K., Hyndman, R.: Characteristic-based clustering for time series data. Data Min. Knowl. Disc. **13**, 335–364 (2006)
132. Werner, G., Yang, S., McConky, K.: Time series forecasting of cyber attack intensity. In: Proceedings of the 12th Annual Conference on Cyber and Information Security Research, pp. 1–3 (2017)
133. Wu, W., He, L., Lin, W., Su, Y., Cui, Y., Maple, C., Jarvis, S.: Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality. IEEE Trans. Knowl. Data Eng. **34**(9), 4147–4160 (2020)
134. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.I.: Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, pp. 117–132 (2009)
135. Yasasin, E., Prester, J., Wagner, G., Schryen, G.: Forecasting it security vulnerabilities-an empirical analysis. Comput. Secur. **88**, 101610 (2020)
136. Yu, Y., Zeng, X., Xue, X., Ma, J.: Lstm-based intrusion detection system for vanets: a time series classification approach to false message detection. IEEE Trans. Intell. Transp. Syst. **23**(12), 23906–23918 (2022)
137. Yuan, H., Li, H.: Time series intrusion warning with gan for missing data in cps. In: Proceedings of the 2023 11th International Conference on Communications and Broadband Networking, pp. 59–64 (2023)
138. Zängerle, D., Schiereck, D.: Modelling and predicting enterprise-level cyber risks in the context of sparse data availability. The Geneva Papers on Risk and Insurance-Issues and Practice **48**(2), 434–462 (2023)
139. Zhang, T., Qiu, H., Castellano, G., Rifai, M., Chen, C.S., Pianese, F.: System log parsing: a survey. IEEE Trans. Knowl. Data Eng. (2023)
140. Zhang, Z., Mal, C., Ding, B., Gao, M.: Detecting manipulated facial videos: A time series solution. In: 2020 25th International Conference on Pattern Recognition (ICPR), pp. 2817–2823. IEEE (2021)
141. Zhang Wu, M., Luo, J., Fang, X., Xu, M., Zhao, P.: Modeling multivariate cyber risks: deep learning dating extreme value theory. J. Appl. Stat. **50**(3), 610–630 (2023)

142. Zhao, N., Jin, P., Wang, L., Yang, X., Liu, R., Zhang, W., Sui, K., Pei, D.: Automatically and adaptively identifying severe alerts for online service systems. In: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, pp. 2420–2429. IEEE (2020)
143. Zhou, P., Wang, Y., Li, Z., Wang, X., Tyson, G., Xie, G.: Logsayer: Log pattern-driven cloud component anomaly diagnosis with machine learning. In: 2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS), pp. 1–10. IEEE (2020)
144. Zhou, P.Y., Chan, K.: A model-based multivariate time series clustering algorithm (2014). https://doi.org/10.1007/978-3-319-13186-3_72

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.