Infrastructure anomaly detection: A cloud-native architecture at Germany's Federal Employment Agency

Gebhard Herget[1], Eldar Sultanow[2], Alina Chircu[3], Johannes Ludsteck[4], Sebastian Hammer[5], Christian Koch[6], Willy Reuter[7], Matthias Seßler[8]

Abstract: In prior research we explored the use of time series analysis methods to detect one class of information technology (IT) infrastructure anomalies - Distributed Denial of Service (DDoS) attacks. The results of this prior work were a mathematical model and a prototype implementation that were concretely trialed and operated in the data centers of Germany's Federal Employment Agency (FEA). With this paper, we go one step further and generalize as well as optimize the mathematical model and create higher performance and scalability for an updated prototype through targeted use of cloud technologies. The starting point of our generalization is the Exponential Smoothing (E-S) approach, which underlies, for example, the well-known Holt-Winters method. This method is used to predict univariate time series. To detect anomalies (such as DDoS attacks) in infrastructure data, we extend the E-S approach to enable it to forecast multivariate time series. In this optimization of our method and our prototype, we take an exploratory, agile approach. Furthermore, we present a cloud-native architecture stack which we pilot in Azure.

Keywords: Time Series; Enterprise Architecture; Exponential Smoothing; Cloud; Azure; Distributed Denial of Service; Infrastructure Anomaly Detection System

1 Introduction

The availability, integrity and security of an organization's information technology (IT) infrastructure are paramount for providing IT services internally and externally to various organizational stakeholders such as employees or customers. IT infrastructure anomalies, i.e. deviation from normal system behavior that result from outside malicious threats, internal errors, or atypical usage [Fe19], are prime concerns for companies to identify

[1] IT Dept. of Germany's Federal Employment Agency, Tafelhofstraße 4, 90443 Nuremberg, Germany gebhard.herget@arbeitsagentur.de
[2] Capgemini, Bahnhofstraße 30, 90402 Nuremberg, Germany eldar.sultanow@capgemini.com
[3] Bentley University, 175 Forest Street, Waltham, MA 02452-4705, USA achircu@bentley.edu
[4] Institute for Employment Research (IAB), Regensburger Straße 104, 90478 Nürnberg, Germany johannes.ludsteck@iab.de
[5] Capgemini, Bahnhofstraße 30, 90402 Nuremberg, Germany sebastian.hammer@capgemini.com
[6] Technische Hochschule Nürnberg Georg Simon Ohm, Kesslerpl. 12, 90489 Nuremberg, Germany christian.koch@th-nuernberg.de
[7] Microsoft Deutschland GmbH, Unter den Linden 17, 10117 Berlin, Germany willy.reuter@microsoft.com
[8] IT Dept. of Germany's Federal Employment Agency, Tafelhofstraße 4, 90443 Nuremberg, Germany matthias.sessler@arbeitsagentur.de

doi:10.18420/inf2022_101

and address. Distributed Denial of Service (DDoS) attacks are an especially important and widespread type of malicious attack towards a company's IT infrastructure. Internet security reports indicate there were 9.7 million DDoS attacks in 2021 - which represents a slight decrease from 2020 numbers (likely a result of the COVID-19 global pandemic) but is significantly higher than 2019 DDoS numbers [NE22]. DDoS attacks are cheap to launch (sometimes even free), increasingly target specific industries such as voice over IP, computer manufacturing, software publishers, insurance agencies and brokerages, and colleges, universities and professional schools, and generate significant economic losses - sometimes to the tune of tens of millions of dollars - for the affected companies [NE22].

In prior work by Ludsteck et al. [Lu21], we proposed an architecture to detect anomalies in the IT infrastructure and described the implementation of a prototype which was tested in the context of detecting Distributed Denial of Service (DDoS) attacks for the data centers of Germany's Federal Employment Agency (FEA). Since then, the prototype has been in use for about a year, yielding additional insights regarding both technologies and data analysis. In this paper, we build on the lessons learned from the prototype and present an optimized architecture for DDoS anomaly detection at FEA. The contributions of this paper are twofold: the generalization of the time series analysis method used for IT infrastructure anomaly detection and the cloud-enabled scalable architecture. With respect to the first point, we show how the proprietary time series analysis implementation can be replaced with a generalized multivariate exponential smoothing method. With respect to the second point, we discuss a trial of a cognitive cloud service from Azure.

2 Related Work

Our prior work reviews the main characteristics of IT infrastructure anomalies and the related anomaly-based detection systems [Lu21]. Other authors have offered comprehensive surveys of these issues (see, for example, [Fe19]). In this section, we review additional work relevant to the design of the anomaly detection system - namely to the prediction of abnormal IT infrastructure events through time series, which will be used to optimize our prototype.

He et al. [He20] address the limitation of many existing studies on time series prediction, which consider temporal patterns without regard to correlation among variables, thus causing loss of information and inevitably producing false positives. The authors design an unsupervised method to detect anomalies in multivariate time series, which consists of two steps – a prediction step and a threshold selection step. In the first step, they utilize a multi-scale convolution and graph attention network for capturing information in temporal pattern with feature pattern. In the latter step, they identify the threshold value by an MSE (mean square error) based extreme value analysis between the predicted and the observed value.

Li et al. [LPJ17] introduce a framework for anomaly detection in multivariate time series that is based on Hidden Markov Models. Their main idea is to convert multivariate time series into univariate ones. For this, the authors consider conversion algorithms that include fuzzy C-Means (FCM) clustering and fuzzy integral. They apply a Hidden Markov Model (HMM) to discover the presence of anomalies in multivariate time series.

In related work, Li et al. [Li20] present an approach using fuzzy clustering technique to reveal anomalies with respect to the amplitude and shape of multivariate time series. In the first step, the authors use a sliding window in order to construct an array of multivariate sub-sequences. In the second step, they employ an enhanced fuzzy clustering for finding a pattern within these obtained sub-sequences. In the third and final step, they use a reconstruction criterion to recreate the sub-sequences involving the optimal cluster centers and partition matrix. Using a confidence index, they quantify the degree of any anomaly found and tune the detection itself via particle swarm optimization. The authors successfully validate their approach using experiments on various synthetic and genuine data sets.

Jones [Jo66] develops an exponential smoothing method for multivariate time series. His method recursively "estimates the optimum weight matrix for the exponential smoothing and prediction of multivariate time series" and "generalizes to non-linear processes when the non-linear structure is known" [Jo66].

Pfeffermann and Allon [PA89] introduce a "multivariate extension of the familiar exponential smoothing procedure for forecasting univariate time series composed of level, seasonality and irregularity". Using an univariate smoothing procedure together with corrective factors relying upon information from the other series, the authors deduce the up-to-date level, trend, and seasonal effect estimates of each series as weighted estimate averages. Moreover, they use two actual bivariate time series to demonstrate and benchmark the procedure's performance against that of other univariate and multivariate prediction methods.

Zhao et al. [Zh20] address one of the current major limitations of anomaly detection on multivariate time-series: "they do not capture the relationships between different time-series explicitly, resulting in inevitable false alarms". To close this gap, the authors develop a new self-supervised method that recognizes anomalies in multivariate time series by treating each univariate time series as a separate feature and by simultaneously incorporating two graph attention layers designed to learn the (non-trivial) interdependencies between multivariate time-series across both temporal and feature dimensions.

3 Using A Cognitive Time Series Service with Recurrent Neural Network for Infrastructure Anomaly Detection

The work of Zhao et al. [Zh20] forms the basis of Microsoft's Cognitive Time Series service. We used this service to optimize the architecture described in Ludsteck et al. [Lu21]. We

fed our FEA infrastructure data to the service using the Azure Data Explorer[9], which is part of Microsoft Azure Cloud.

For this purpose, first an Azure Data Explorer cluster had to be setup on the Azure Cloud. This allowed the capacity of the system to be expanded dynamically, as new instances could be created on the fly. This is particularly advantageous for later productive usage. Furthermore, the system included a database for both historical data as well as for newly arriving real-time data.

The Azure Data Explorer offers various options for feeding data into the system. One is to stream real-time data via an Event Hub or IoT Hub. In addition, the upload of historical data via file upload is possible. Since we did not have a connection to a system from which data can be streamed yet, we used the file upload. For this, the data was brought into a file format supported by Azure Data Explorer. We chose .csv for this purpose. The converted file could then be uploaded via the Data Explorers web interface.

The Data Explorer offers the option to prepare or plot the data using KQL (Kusto Query Language)[10]. Listing 1 shows the code for extracting anomalies from the data. Thereby, the considered time period was defined with min_t and max_t and the time interval was specified with 10 minutes. The anomalies could then be determined by using the function *series_decompose_anomalies* and then plotted using an anomaly chart. As parameters we used the calculated Mahalanobis distance (a multivariate distance metric) and a anomaly threshold of 7.0.

```
let min_t = datetime(2022-06-19 22:00);
let max_t = datetime(2022-06-20 23:00);
let dt = 10m;
monitoring_data
| make-series num=avg(['mahalanobis']) on Timestamp from min_t to max_t step dt
↪  by Country
| where Country == "*****"
| extend (anomalies, score) = series_decompose_anomalies(num, 7.0, -1, 'linefit')
| render anomalychart with(anomalycolumns=anomalies)
```

Listing 1: KQL (KUSTO Query Language) Query for Azure Data Explorer

An example of anomalies determined by the Azure Data Explorer is shown in Figure 1 with anomalies indicated by red dots. The blue line shows the Mahalanobis distance and the red line the calculated anomaly score. A total of four anomalies were detected when the threshold was set to 7.0. These anomalies correspond in time to the alerts shown in Figure 5.

[9] https://azure.microsoft.com/de-de/services/data-explorer/
[10] https://docs.microsoft.com/de-de/azure/data-explorer/kusto/query/

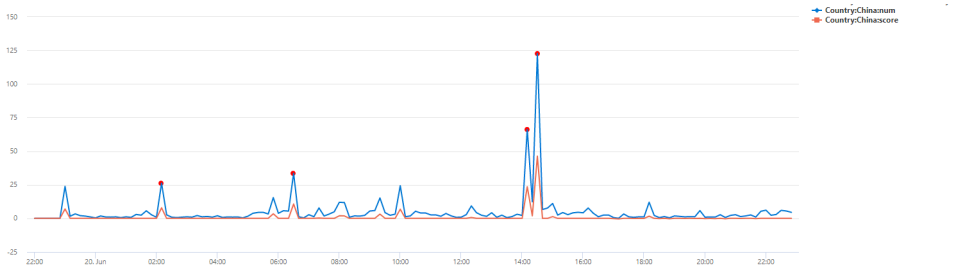

Fig. 1: Anomaly detection graph by Azure Data Explorer (Anomalies are marked with red dots)

Our approach is also capable of running in the Microsoft Azure Cloud. The next steps would be to connect the real-time system to the Azure Data Explorer using the Event Hub and to implement automatic notifications when anomalies are detected.

4 Architectural Findings from the Prototype

The prototype has now been running for almost one year and we gained valuable lessons from its operation.

One of these lessons is that **the Mahalanobis distance is very suitable to detect outliers in the multidimensional time series data of DDoS infrastructure anomalies.** Kotu and Deshpande [KD18] indicate that "[m]ore advanced statistical techniques take multiple dimensions into account and calculate the Mahalanobis distance instead of the standard deviations from the mean in a univariate distribution". Indeed, the "Mahalanobis distance is the multivariate generalization of finding how many standard deviations away a point is from the mean of the multivariate distribution" [KD18]. To detect cyber-attacks on water distribution system, Gjorgiev and Gievska [GG20] employ deep learning architectures based on Variational Autoencoders using Mahalanobis distance as well.

But why is Mahalanobis distance so helpful in our case? The current problem with monitoring is that metrics are surveilled using static thresholds. These thresholds (limit values) must be defined upfront, entered into a monitoring tool accordingly, and then assigned to the metrics/components. The definition of these limit values can be complicated in several ways. In the case of technical threshold values such as CPU temperature or the fill level of a file system, they are usually still easy to determine. For more complex metrics such as the number of accesses to a service or permissible error rates, determining the threshold values can be much more difficult. In addition, threshold values are often dynamic and depend on other metrics.

For this purpose, the Mahalanobis distance method was used as a solution. The metrics used by our prototype were based on aggregates from access logs over a defined time interval of 10 minutes (6 measure points per hour) each.

- *awz*: Response time (mean value)
- *awz.std_dev*: Variance of the response time
- *req*: Number of accesses (requests)
- *ips*: Number of clients (unique IPs)
- *bytes*: Bytes transferred

The method was initially applied to historical data in order to determine its suitability based on existing logs. The analysis was based on data sets of one day in each case. Thus, 5 metrics with $6 \cdot 24$ measurement points were analyzed. The metrics had further attributes such as response codes (200, 404, 500 etc.) or observed services. With 30 services and approx. 30 response codes and further attributes, this resulted in up to: 5 metrics · 30 vhosts · 30 response codes · 200 countries (sources) = 900, 000 time series per day.

For the 5 mentioned metrics the Mahalanobis distance was calculated. This resulted in 30 vhosts · 30 response codes · 200 countries (sources) = 180, 000 attribute combinations. Note that this is the worst case and in practice we handled about 5000 attribute combinations. For each of these 5000 attribute combinations, 5 time series are available.

Using three nested for-loops (over *vhost*, *response_code*, *country*) we went through each attribute combination and considered one time series, for example: all cases where $response\_code = 200$ and the request originated from Italy targeting service "X". And from this time series we picked out the outliers by applying the Mahalanobis distance. For an alert, the statistical significance of each value was determined. The p-value for each distance was calculated as the p-value corresponding to the chi-squared statistic of the Mahalanobis distance with $k - 1$ degrees of freedom, where k is the number of variables. In our case, we used $5 - 1 = 4$ degrees of freedom.

Then warnings were triggered at a Mahalanobis distance of more than 25 and a p-value lower than 0.001. These warnings were summed up on the time axis and if more than 10 alarms were exceeded at the same time, a problem was flagged.

The parameters described above were determined based on the evaluation of different scenarios and are subject to further optimization in the future. The result is a strongly condensed representation of anomalies, which facilitates a visual evaluation via a time series plot. Parameterization is only required at a few points, which allows monitoring of such systems with reasonable effort. Retrospectively, it was possible to detect an anomaly that was not flagged by the currently installed safety mechanisms. A further optimization of our prototype lies in the smoothing of the time series (by using a Kalman filter for example).

In practice, how does the DDoS detection work? Figure 2 shows the result of aggregation over 10 minutes intervals of the utilization data containing the *vhost* and *country*. That

means all requests from all *countries (sources)* on all *vhost* (5000 attribute combinations) are shown in this example given by Figure 2. This specific example is from 2022-06-20 to 2022-06-21.

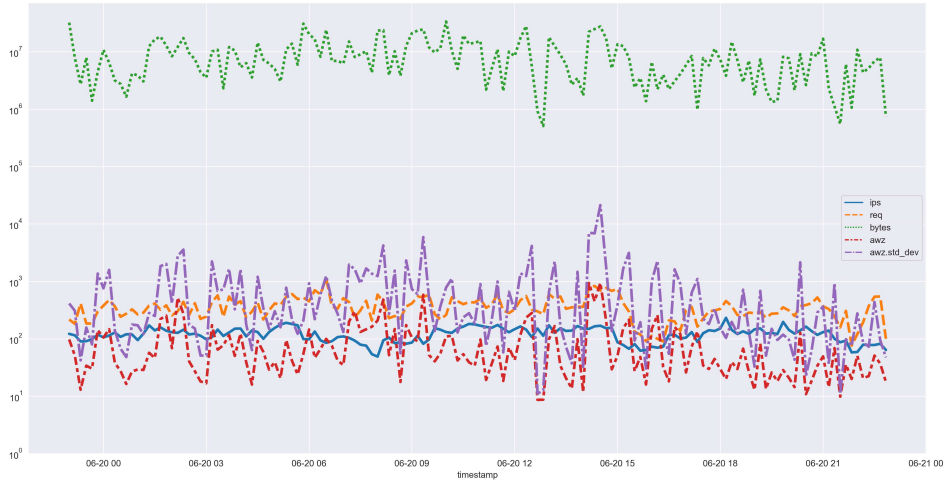

Fig. 2: Aggregation (10 minutes intervals) of of the utilization data containing the *vhost* and *country* (20th to 21st of June 2022)

Figure 3 shows the same raw data as shown in Figure 2. But Figure 3 additionally contains the calculated values of the Mahalanobis distance between the 5 time series (ips number of unique IP addresses, req number of requests, $bytes$ number of bytes transferred, awz response time, $awz.std\_dev$ standard deviation of response time) depicted by Figure 2. This Figure 3 shows the Mahalanobis distance plot as the pink curve, the p curve as the grey curve (the p curve is the Chi-squared test curve - which we obtain by applying Chi square Test on the Mahalanobis distance curve), the $alert$ curve is the result of filtering the p curve together with the Mahalanobis distance curve (the p value must be lower than 0.001 and the Mahalanobis distance value must be larger than 25).

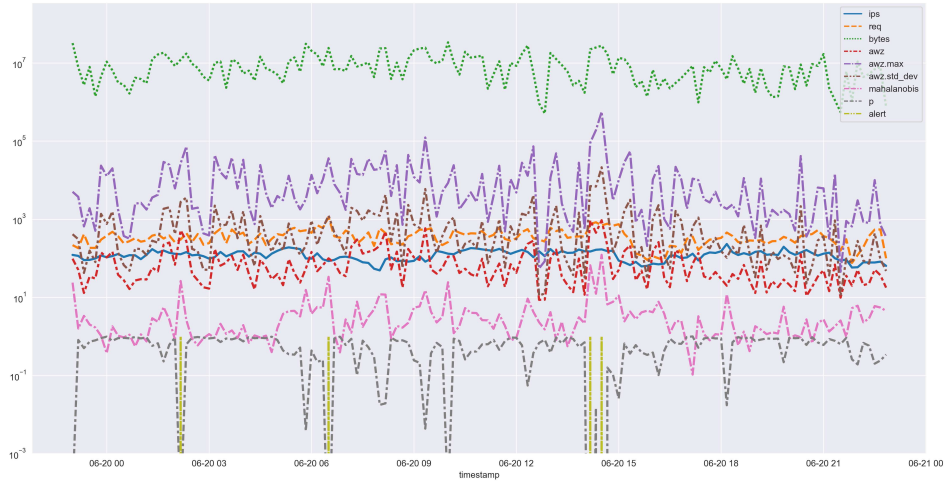

Fig. 3: Example from Figure 2 extended by Mahalanobis Distance curve, p curve and identified alerts.

To get a clearer view, Figure 4 shows only the Mahalanobis distance curve, p curve and identified alerts, which we added to Figure 3.

Fig. 4: only the Mahalanobis distance curve, p curve and identified alerts, which we added to Figure 3 (20th of June 2022)

Figure 5 shows a pair plot (scatterplot matrix) over the five parameter values ips (number of unique IP addresses), req (number of requests/accesses), $bytes$ (number of bytes transferred), awz (response time), $awz.std\_dev$ (standard deviation of response time). The outliers – which are the alerts we identified (by p value being lower than 0.001 and the

Mahalanobis distance larger than 25) – are highlighted in orange. And these orange outliers refer to the (downward) peaks in Figure 3 and Figure 4.

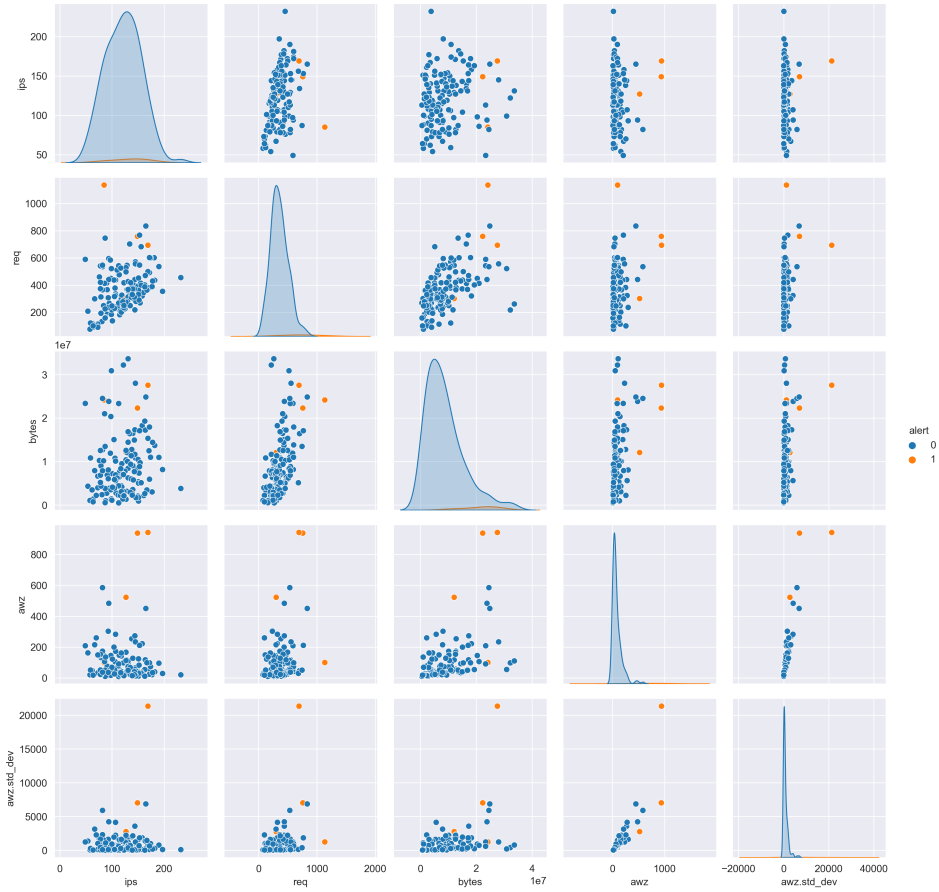

Fig. 5: Pair plot / Scatterplot matrix over the 5 paramter values (ips, req, $bytes$, awz, $awz.std\_dev$)

Figure 6 accumulates all identified alerts in order to determine wether or not there is any prominence in the density of the anomalies within the considered/observed time period (in this case, from 20th to 21st of June 2022). In this plot given by Figure 6 we see two prominent peaks worth taking a closer look at, which we do in Figure 7.

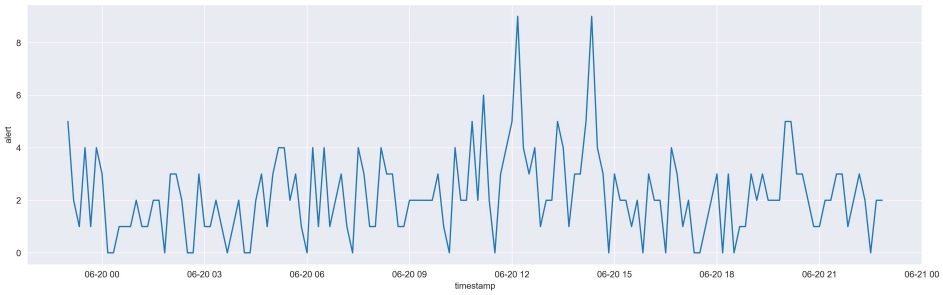

Fig. 6: All identified alerts accumulated

Figure 7 shows, in a heat map, a prominent line in the middle of the plot. This line indicates the countries involved in these important anomalies.

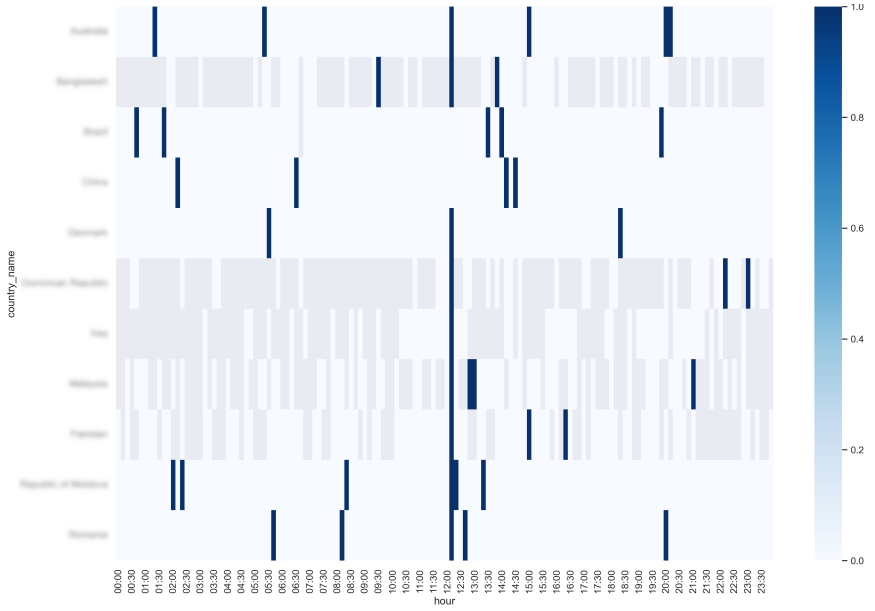

Fig. 7: Heat map of countries (sources) from which the requests originate

5 Prototype Multivariate E-S Method and its Architecture

In our previous prototype, we proposed our own method for time series analysis (see [Lu21]). We developed this original method to model the differently chosen minute-lags as features because established methods such as the Holt-Winters method were designed for univariate time series. In our optimized prototype, we took a step towards standardization by going back

to the well-established Holt-Winters method and making it able of handling multivariate time series. Listing 2 shows the corresponding code. This method is computationally intensive and therefore we anticipate running it in the cloud.

```

import datetime
begin_time = datetime.datetime.now()
mya = np.array([[0.15, 0.15], [0, 0.15]])
myb = np.array([[0.15, 0.15], [0, 0.15]])
myc = np.array([[0.15, 0.15], [0, 0.15]])

def multivariate_ES (data: np.array, h_period: int, A: np.array, B: np.array, C:
↪  np.array, m: int, year_size: int):
    L = []
    T = []
    data = data[-h_period:, :]
    inits = initialize (data,year_size)
    L.append (inits[0])
    T.append (inits[1])
    S = []
    for i in range (inits[2].shape[0]):
        S = np.append ([S], inits[2][i,:])
    shape = (int(S.shape[0]/2), 2)
    S = S.reshape(shape)
    s = []
    for i in range (inits[2].shape[0]):
        s = np.append ([s], inits[2][i,:])
    shape = (int(s.shape[0]/2), 2)
    s = s.reshape(shape)

    for i in range (h_period-1):
        L.append ((A @ (data[i+1,:] - S[i % year_size])) + ((np.identity(2) - A)
        ↪  @ (L[i]+T[i])))
        T.append ((B @ (L[i+1] - L[i])) + ((np.identity(2) - B) @ T[i]))
        S [(i+1)
        Smean = S.mean(axis = 0)
        for j in range(year_size):
            S [j] = S[j] - Smean
        s = np.append ([s], (S [(i+1) % year_size]))
        shape = (int(s.shape[0]/2), 2)
        s = s.reshape(shape)
    mves = []
    for i in range(6):
        mves.append(L[-1] + (i+1)*T[-1] + S[(h_period-1) % year_size])
    return pd.DataFrame(L), pd.DataFrame(T), pd.DataFrame(s), mves
```

Listing 2: Our Multivariate E-S Method

6 Conclusions and Future Research

In this paper, we describe a time series method and a cloud-native architecture for detecting DDoS attacks - which are one particularly important type of IT infrastructure anomalies in companies today. Our prototype was trialed in the data centers of Germany's Federal Employment Agency (FEA), and we used insights from its operation over the past year to optimize it and plan future refinements.

One area of improvement is the alert capability of the system. We plan to implement a live/real-time alert. This will alert users not only based on retrospective analysis, but also by detecting anomalies during operation and flagging them immediately, as well as by forecasting future anomalies. Automatic report generation will also be included along other commodity functions. In addition, we plan to differentiate more precisely which alert we are dealing with (compared to the current alert function which just identifies a generic alert). The system should be able to determine if an anomaly is really DDoS-related or rather something due to internal service level losses. We can achieve this by further refining the feature engineering. For example, response time is more of a service level feature and worth considering if that should be included for DDoS detection.

Another area of improvement is developing a true real-time architecture. We have demonstrated the Azure prototype in Section 3. The next step is to conceptualize and implement the full data pipeline into Azure such that computation and identification of anomalies will be completed in (almost) real-time. A follow-up step is to optimize the sample rate and, if necessary, define the reference time series for evaluation of individual measurements. Yet another future step is the conversion to real-time using containers according to FEA standards. We also plan to check raw data for usability (classifying whether evaluation is possible). Moreover we want to automatically perform analysis on any elevated anomaly density and document it, in order to support rapid root cause analysis. Additional improvements include classification of anomalies (e.g. technical failure external, internal, crawler access, etc.) and supporting the classification by deriving further metrics from the log data and analyzing them.

This work contributes to the growing body of literature regarding anomaly detection for IT infrastructure, in general, and DDoS attacks, in particular. We anticipate a continuing need for research in these areas to refine the existing anomaly detection methods and systems and support and protect increasingly complex IT infrastructures for digital transformation, internet of things, and other emerging IT trends.

Literatur

[Fe19] Fernandes, G.; Rodrigues, J. J.; Carvalho, L. F.; Al-Muhtadi, J. F.; Proença, M. L.: A comprehensive survey on network anomaly detection. Telecommunication Systems 70/3, S. 447–489, 2019.

[GG20] Gjorgiev, L.; Gievska, S.: Time Series Anomaly Detection with Variational Autoencoder Using Mahalanobis Distance. In (Dimitrova, V.; Dimitrovski, I., Hrsg.): ICT Innovations 2020: ICT Innovations 2020. Machine Learning and Applications. Bd. 1316. Communications in Computer and Information Science, Springer, S. 42–55, 2020, ISBN: 978-3-030-62097-4.

[He20] He, Q.; Zheng, Y. J.; Zhang, C.; Wang, H. Y.: MTAD-TF: Multivariate Time Series Anomaly DetectionUsing the Combination of Temporal Pattern and Feature Pattern. Complexity 2020/, 2020.

[Jo66] Jones, R. H.: Exponential Smoothing for Multivariate Time Series. Journal of the Royal Statistical Society 28/1, S. 241–251, 1966.

[KD18] Kotu, V.; Deshpande, B.: Data Science: Concepts and Practice. Morgan Kaufmann, 2018, ISBN: 978-0-12-814761-0.

[Li20] Li, J.; Izakian, H.; Pedrycz, W.; Jamal, I.: Clustering-based anomaly detection in multivariate time series data. Applied Soft Computing Journal 100/, 2020.

[LPJ17] Li, J.; Pedrycz, W.; Jamal, I.: Multivariate Time series Anomaly Detection: A Framework of Hidden Markov Models. Applied Soft Computing 60/, S. 229–240, 2017.

[Lu21] Ludsteck, J.; Sultanow, E.; Chircu, A.; Herget, G.; Seßler, M.: An architecture for detecting infrastructure anomalies at Germany's Federal Employment Agency. In: INFORMATIK 2021 – Computer Science & Sustainability. Gesellschaft für Informatik, 2021.

[NE22] NETSCOUT: Threat Intelligence Report Issue 8: Findings from 2nd Half 2021./, 2022.

[PA89] Pfeffermann, D.; Allon, J.: Multivariate exponential smoothing: Method and practice. International Journal of Forecasting 5/1, S. 83–98, 1989.

[Zh20] Zhao, H.; Wang, Y.; Duan, J.; Huang, C.; Cao, D.; Tong, Y.; Xu, B.; Bai, J.; Tong, J.; Zhang, Q.: Multivariate Time-Series Anomaly Detection via Graph Attention Network. In: 2020 IEEE International Conference on Data Mining (ICDM). IEEE, 2020.