Slovak University of Technology
Faculty of Informatics and Information Technologies

Ing. Lenka Beňová

Dissertation Thesis Abstract

Increasing the security of web systems: Log anomaly detection

to obtain the Academic Title of
"philosophiae doctor", *abbreviated* as "PhD."

in the doctorate degree study programme:
Applied Informatics (2511V00)

in the field of study:
Computer Science

Form of Study:
full-time

Place and Date:
Bratislava, 27.11.2023

STU

Dissertation Thesis has been prepared at:
Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Slovakia

Submitter: Ing. Lenka Beňová
Institute of Computer Engineering and Applied Informatics
Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 2, 842 16 Bratislava

Supervisor: doc. Ing. Ladislav Hudec, CSc.
Institute of Computer Engineering and Applied Informatics
Faculty of Informatics and Information Technologies, Slovak University of Technology
Ilkovičova 2, 842 16 Bratislava

Dissertation Thesis Abstract was sent: .....

Dissertation Thesis Defence will be held on .....

at..... (am/pm) at .....

.....

prof. Ing. Ivan Kotuliak, PhD.
Dean of the Faculty of Informatics and Information Technologies

Abstract:

System logs primarily document significant events and conditions, especially during critical periods. These logs are essential for both online monitoring and anomaly detection, as they give a detailed picture of a system's status during critical moments. Traditional anomaly detection methods often miss new or unexpected behavior patterns. This highlights the need for machine learning-based approaches that can better identify such anomalies, making the analyst's job more straightforward and automated.

The focus of this dissertation is to study state-of-the-art machine learning methodologies in network anomaly detection. The objective was to devise a new methodology that applies machine learning techniques for anomaly detection in large web server log datasets. This approach takes into account both individual user behavior and the broader traffic patterns. The main goal is to effectively use machine learning algorithms for anomaly detection using the gathered HTTP log data, with a clear focus on automation and real-world application with extensive data inputs.

Abstract in Slovak Language:

Primárnym účelom systémových logov je zachytiť dôležité udalosti a stavy počas kritických momentov, aby pomohli zistiť príčinu systémových útokov, výpadkov či iných anomálií. Tieto logy sú dôležité nielen pre online sledovanie, ale aj pre detekciu anomálií, keďže poskytujú podrobný pohľad na stav systému počas týchto kľúčových momentov. Klasické metódy na detekciu anomálií často nedokážu rozpoznať nové alebo neočakávané vzory správania. To zdôrazňuje potrebu metód založených na strojovom učení, ktoré by tieto anomálie dokázali lepšie identifikovať, a tým zefektívniť a automatizovať prácu analytikov.

Táto dizertačná práca sa zameriava na štúdium najnovších metodológií strojového učenia v oblasti detekcie sieťových anomálií. Cieľom práce bolo navrhnúť novú metodológiu, ktorá využíva techniky strojového učenia na detekciu anomálií v rozsiahlych datasetoch logov webových serverov. Tento prístup berie do úvahy správanie jednotlivých užívateľov aj celkové vzory premávky. Hlavným zámerom je efektívne používať algoritmy strojového učenia na detekciu anomálií na základe zhromaždených HTTP logov, s dôrazom na automatizáciu a praktickú aplikáciu v reálnych podmienkach s veľkým množstvom dát.

STU

Table of Contents

STU

Introduction

The amount of harmful network traffic is continually increasing. Automatic data processing and intrusion detection systems based on machine-learning algorithms that can detect threats before they become noticeable are in high demand. Because understanding each log entry from a webserver request requires a high level of ability, and even more so to understand sequences of actions like continuous web requests, there is a strong motivation to create systems that can detect both known and unknown intrusions.

The number of anomalies generated by various network misconfigurations continues to rise as network traffic grows, resulting in more successful network attacks. To protect the confidentiality, availability, and integrity of computer systems, network anomalies must be recognized and analyzed; as a result, intrusions consume resources and bandwidth, leaving network services unavailable.

Numerous attackers on the Internet are continually attacking critical computer systems, and intrusion detection systems (IDSs) are critical in defending them. Attacks are addressed using signature-based techniques, which extract important characteristics and create a unique signature for each attack. These methods are extremely successful against previously captured attacks. They do not, however, have the ability to detect novel intrusions or zero-day attacks, and they are not ideal for real-time anomaly detection across huge datasets (Ni, 2016), (Iwan Syarif, 2012).

Network intrusion detection systems (NIDSs) are security systems that monitor harmful activities in network traffic and create alerts when suspicious activity is discovered so that the cause of the alert can be investigated and action taken. Traditional ways to network anomaly detection are becoming ineffective as network attacks become more complex due to technological improvements (Monowar H Bhuyan, 2017), (Naveed Chouhan, 2019)

We can detect if a specific port or service on a specific machine or machines is being connected to or transacted with at an abnormal rate using anomaly detection, implying that there is some kind of intrusion activity going on where some intruder is attempting to hack into the specific system or systems. This is highly useful information for the operations team, who can rapidly call in cybersecurity experts to try to figure out what's really going on and take any preventive or proactive measures rather than reactivating the system (Adari., 2019)

Anomaly detection in networks is based on locating data that does not follow predictable patterns. Despite the availability of different approaches, there are still a number of research challenges. Data contains noise, which is an abnormality in and of itself, making it difficult to differentiate. As a result, there is no generally applicable anomaly detection approach. Because

intruders are aware of present procedures, there is a lack of publicly available labelled datasets, necessitating the development of more complicated and novel approaches.

Various approaches to anomaly detection have been developed, however most of them have drawbacks when utilized in real-world circumstances.

This research intends to show how machine learning may be used to discover anomalies in a variety of application servers that provide modular updates to customers throughout the world, with a focus on automation and usability in a real-world context.

Related work

The main goal of logs is to capture states and significant events at various critical moments to aid in system error troubleshooting and root cause analysis. This kind of data is accessible in almost all computer systems. Since logs are a significant and valuable resource for understanding the system state and performance issues, various system records are good sources of information for online monitoring and anomaly detection.

Researchers Naseer (Naseer, 2018), Malaiya et al. (Malaiya, 2018), and Kim et al. (Kim, 2018) investigated the potential of deep neural networks for intrusion and anomaly detection. Naseer and his team used various deep neural networks and compared their performance with traditional machine learning techniques, achieving an accuracy of up to 89%. Malaiya and colleagues focused on empirical evaluation of deep learning, specifically the LSTM Seq2Seq structure, which achieved a binary classification accuracy of 99% on two traffic data sets. Kim et al. introduced the C-LSTM architecture for anomaly detection in web traffic and showed that it surpasses traditional machine learning techniques with an accuracy of 98.6% on a renowned dataset.

Time Series Prediction

In the field of network communication anomaly detection, time series prediction is key. Models are trained based on past data to predict future network traffic. In most experiments, a combination of LSTM neural network and Discrete Wavelet Transformation was used.

NLP

In the field of Natural Language Processing (NLP), N-grams are often used to extract features from logs. Du et al. (Du, 2017) introduced DeepLog, which uses LSTM networks and surpasses traditional N-grams in encoding complex patterns and long-term states. This system, tested on a large Amazon EC2 server database, was able to detect anomalies at the log record level and showed better performance than other methods. Bertero et al. (Bertero, 2017) approached log analysis using NLP tools and word embedding methods based on word2vec, achieving a 90% accuracy in detecting stress patterns. Vartouni et al. (Vartouni, 2018) used the n-gram model in combination with the SAE method to detect malicious HTTP requests in the CSIC 2010 database, demonstrating their representative power with vector features of various dimensions.

LSTM

Torres et al. (Torres, 2017) used LTE probes from the MONROE project to predict future network behavior. They tried two approaches: the ARIMAX model and a naive model. The data for their experiments came from Lisbon, Portugal. It turned out that the second approach was more accurate. Huo et al. (Huo, 2019) created the LTS-TP model for network traffic prediction, taking into account periodicity and long-term relationships in the data. They used STL decomposition combined with the Seq2Seq model with an enhanced attention mechanism. They tested on a public dataset from MAWI and found that LTS-TP is more effective in predictions. Radford et al. (Radford, 2018) used the ISCX IDS dataset for network attack detection. They applied LSTM RNN and successfully detected abnormal network activity. Casado-Vara et al. (Casado-Vara, 2021) developed an architecture for predicting web traffic using LSTM, using data from Wikipedia. Their model provided accurate predictions. Zhao et al. (Zhao W. a., 2021) proposed a method combining EMD and LSTM for network traffic prediction. They used data from a private ISP and achieved good results with the EMD-LSTM method.

Wavelet Transformation

Wavelet transformation helps identify frequency domain components in time series data, thereby increasing the accuracy of predictive models. Shelatkar et al. (Shelatkar, 2020) used DWT on Wikipedia data. They divided it into low and high-frequency segments for optimal processing by ARIMA and RNN models, leading to improved results. Their methodology integrates ARIMA and RNN using DWT. Zhao et al. (Zhao Y. a., 2018), on the other hand, use a wavelet approach in combination with various neural networks to detect frequency details of time series, surpassing seven basic predictive methods in their experiments on stock price data sets and electricity consumption.

Transformer

The Transformer architecture was introduced by researchers from Google Brain and the University of Toronto in the article "Attention is all you need". This architecture bypasses recursion to allow parallel processing, utilizing multiple attentions and positional embeddings. The Transformer processes sequences as a whole, not sequentially, thanks to the use of self-attention. This method measures the similarity between items within a sentence.

Some of the significant features and findings regarding Transformers include:

1. Improved predictive capabilities, as highlighted by several studies.
2. Use of the encoder-decoder framework, where the output is generated based on the encoded input and previous decoder outputs.
3. Challenges associated with the Transformer model, such as quadratic time complexity, high memory usage, and inherent limitations due to its encoder-decoder design.

Xu et al. (Xu, 2021) introduced the Anomaly Transformer, which includes AnomalyAttention with a dual-branch structure. This model emphasizes the difference between normal and anomalous time points using a minmax method.

Wu et al. (Wu, 2020) compared the performance of the Transformer with models such as ARIMA, LSTM, and Seq2Seq. The Transformer showed superior results, especially in terms of the RMSE metric, where it surpassed LSTM and Seq2Seq models using attention mechanisms.

Zhou et al. (Zhou, 2021) addressed the problem of predicting long sequences (LSTF) by introducing the Informer model. Informer overcomes the inherent challenges of the Transformer by introducing the ProbSparse self-attention mechanism and generative decoder. Based on empirical data, Informer increases predictive capabilities for LSTF. Some of the main challenges of the standard Transformer in addressing LSTF include quadratic self-attention computation, memory limitations due to layering layers for long inputs, and slower prediction speeds for longer outputs.

In their studies, the authors compared the performance of Informer with various time series prediction models using different databases. Informer consistently showed increased predictive capacities across all databases, surpassing other neural network models, such as LSTM.

User Behavior Analysis in the Context of Web Server Logs

In the study, authors Gao et al. (Gao, 2017) analyzed web records from BUPT websites. They used data from these records and user activity to develop a user behavior model. They applied two methods to detect network threats based on user behavior. Using entropy as an intrinsic value of the k-means model, they determined the distance of individual users to the cluster center.

Debnath et al. (Debnath, 2018) introduced LogLens, a real-time log analysis tool using machine learning techniques. LogLens identifies anomalies with minimal human interaction and quickly responds to changes in system behavior.

Zhao et al. (Zhao N. a., 2021) compared five commonly used unsupervised learning algorithms for log anomaly detection, including statistical models and deep learning-based methods. The results showed that while these methods are effective on publicly available datasets, they are not consistently effective on real data. In general, deep learning-based methods outperformed traditional statistical methods.

Dataset

The data for this study come from ESET, spol. s r.o., a renowned company specializing in cybersecurity. Their software is regularly updated by automatically downloading virus signatures and software module updates from the company's servers. The frequency of these updates leads to variable and often significant fluctuations in server resource usage. Usage can sometimes be predictable, but global time zones, weekends, and holidays bring unpredictable changes in server usage.

To continuously monitor these changes, the Zabbix monitoring system, which provides real-time monitoring, is used, while NGINX records each client connection requesting updates. Specific characteristics of these records can be found in Table 1.

Attribute	Description
remote_addr	Client IP address
remote_user	authentification
http_x_updateid	license key
time_local	local time in the Common Log Format
http_host	HTTP server host
request_method	HTTP request method
uri	Path to the update being requested
server_protocol	Server protocol version
status	response status
body_bytes_sent	request body length
request_length	request length including header and body
request_time	request processing time in seconds with a millisecond's resolution
http_user_agent	identification of the client originating the request

Table 1: Logged attributes.

In analyzing this dataset, time series methods were used. This included decomposing time-oriented data into its components: seasonal, trend, and residual. To achieve a stationary signal suitable for analysis, trend and seasonality components were removed using differentiation.

The raw data consisted of logs from one application server captured over 24 hours. These data, divided into 1-minute intervals, led to the creation of a dataset with approximately 500,000 samples. Specific extracted attributes from these logs include various server response codes to client requests, aggregated length of each request, the number of requests, and the exact minute each request was made. This data extraction process is summarized in Table 2.

time_local	count	body_bytes_sent	request_length	status_200	status_401	...
17/Nov/2020:00:00:00	31 195	1 943 783 623	19 372 728	13 499	15 644	...
17/Nov/2020:00:10:00	35 491	2 742 201 823	21 980 125	14 939	17 023	...
17/Nov/2020:00:02:00	29 505	1 664 891 188	18 589 392	10 698	15 239	...
17/Nov/2020:00:03:00	29 321	1 618 338 775	18 494 291	7 496	15 812	...
17/Nov/2020:00:04:00	27 740	1 081 497 242	17 820 751	6 362	14 720	...

Table 2: Preprocessed dataset sample..

Next, these processed data were visualized, as can be seen in Figure 1. However, due to the granular nature of the data with a 1-minute interval, selecting traffic samples was complex. For clarity, the data were further aggregated by hours, which improved the visibility of trends.

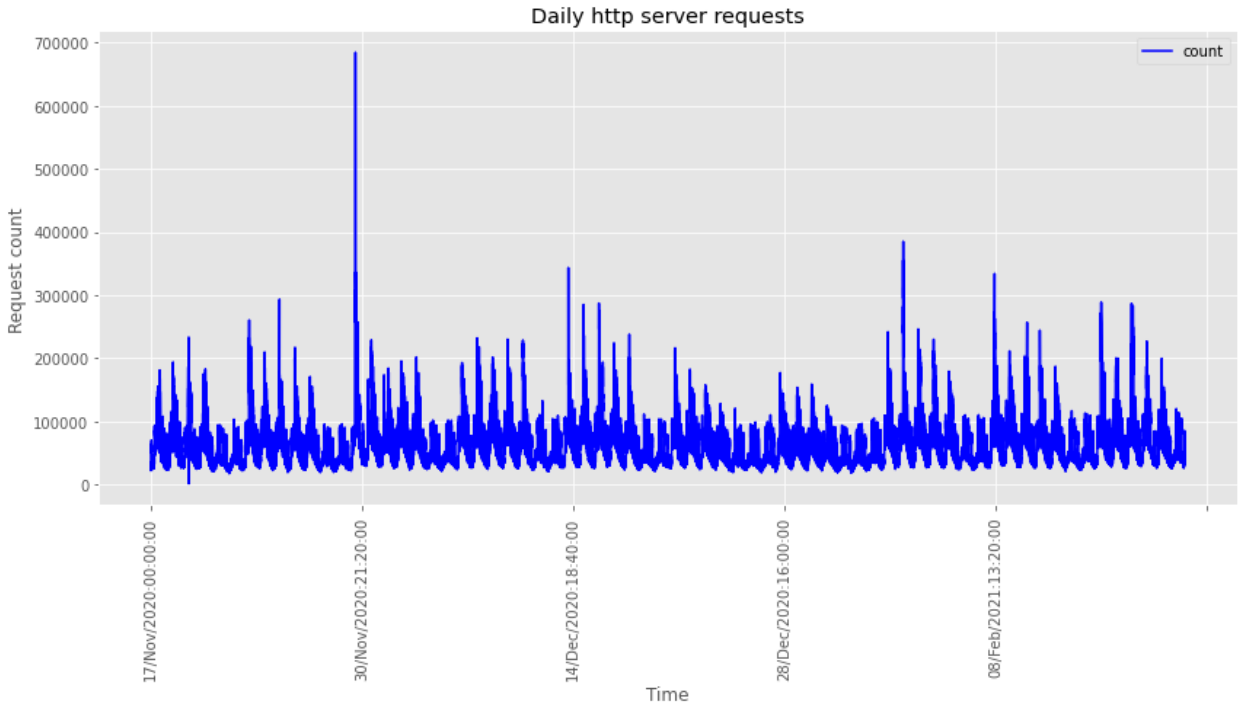

Figure 1: Daily requests visualised.

Another data transformation was performed by differentiating the 'count' value, and its stationarity was confirmed using the Augmented Dickey-Fuller test. Autocorrelation techniques were applied to recognize patterns of repetition in the data. Finally, we attempted to decompose the dataset to clarify the trend, seasonal, cyclical, and residual components, although some efforts, like identifying trends or residuals, proved unsuccessful.

Experiments

In our experiments, we tested two models on our dataset. First, we tried the SARIMA model, which is a specialized version of the ARIMA model with consideration for seasonality. We used 85% of the data for training the model and the remaining 15% for testing. The models were trained on data grouped by minutes and hours, and their performance was visualized in several figures. However, when predicting for the following week, both SARIMA models achieved poor results. Even after optimization using the grid search method, we obtained unsatisfactory results, with an RMSE value of 33602 being too high for practical use.

Subsequently, we tried the XGBoost regressor with 1,000 estimators. Its performance was significantly better, effectively recognizing characteristic patterns in our data. Despite a relatively high error, this model proved to be much better than SARIMA. Moreover, XGBoost identified potential deviations in the data.

STU

Hypotheses

The following hypotheses emerged from the previous analysis of state-of-the-art techniques, analyzed challenges in the field, and characteristics of the collected data.

1. Enhancing the Predictive Capability of Network Traffic Models for Web Servers in the Context of Intrusion Detection.

In this research, we aim to improve the effectiveness of existing models and develop novel methodologies for predicting network traffic patterns on web servers. By harnessing various machine learning techniques and integrating them into a unified system, we endeavor to enhance the prediction process, thereby enabling more accurate intrusion detection.

Our primary focus lies in elevating the accuracy of prediction curves, such as those representing server load-network curves, over time. This heightened precision empowers us to monitor web server characteristics with greater sensitivity. Consequently, we can detect intrusions by identifying discrepancies between the prediction curve and the actual network traffic curve. Parameters like request volume, request sizes, response statuses, and other relevant metrics are integral components of this predictive analysis. The ability to forecast such characteristics is analogous to a time series prediction problem. Subsequently, we will conduct experiments to assess our system's capability in detecting various types of intrusions, not only at the client-side but also encompassing server-side issues, including configuration, hardware, and software errors.

2. Anomaly Detection in User Behavior within a Web Server Environment for Intrusion Detection.

Our research extends beyond system-level modeling to encompass the behavior of individual users within the web server environment. We aim to model user behavior based on their interactions with the system, which can lead to the identification of typical user profiles for the selected web system. Subsequently, we intend to detect intrusions by identifying anomalies within these user profiles, including sudden deviations from established behavior patterns.

This approach allows us to closely monitor and analyze user behavior, especially when it deviates from the norm. If a user's behavior is deemed anomalous, diverging from known user groups, we can take precautionary measures such as restricting their access to the system or implementing honeypot-like responses. The classification of users into groups can be achieved through machine learning methods, both supervised and unsupervised. Given the large volume of data involved, unsupervised methods, including clustering algorithms, appear to be the more suitable choice. Users can be grouped based on factors such as request timings, request frequencies, geographical origins, or specific access patterns.

STU

Prediction of Update Curves

In this section, we utilized an LSTM network to predict the update curve and detect anomalies. We tested our model on publicly available datasets to demonstrate its effectiveness. The dataset for this study contains logs in the format described in Table 1 and are grouped by minutes. The goal of using this dataset is to find potential anomalies that could cause problems and allow us to quickly resolve them.

It's important to note that these data were noisy and contained existing anomalies. To address this issue, we attempted to gather as much data as possible to reduce the amount of anomalies in the overall dataset. We noted that the characteristics of the logs change depending on days and weeks. Therefore, it was necessary to properly account for the time attribute in the data.

The dataset was obtained from several servers from November to February. These servers were located in Bratislava and Vienna and provided us with similar update curves and characteristics.

Our main goal was to predict the number of future requests, so we chose the 'count' attribute as our target output. Due to the variability of data over time, we decided not to use standard library scalers.

Network Architecture

A recurrent neural network was chosen to predict future requests of the update server, as the update curve is based on historical data. By comparing actual values with expected values from the predicted update curve, we were able to assess problems with updates. As seen in Table 3, the network architecture was chosen based on related research and the best predictive results in our studies, where we compared GRU and LSTM in various configurations.

Type	Units	Optimizer	Activation	Loss on train set	Loss on test set
GRU	512	RMSprop	sigmoid	0.0040	0.0041
LSTM	512	RMSprop	sigmoid	0.0039	0.0039
GRU	512	Adam	sigmoid	0.0041	0.0042
LSTM	512	Adam	sigmoid	0.0042	0.0043
GRU	512	RMSprop	ReLU	0.0046	0.0047
LSTM	**512**	**RMSprop**	**ReLU**	**0.0038**	**0.0038**
LSTM	256	RMSprop	ReLU	0.0054	0.0054
LSTM	1024	RMSprop	ReLU	0.0295	0.0295

Table 3: LSTM and GRU comparison.

Update server logs were collected and aggregated to create the input dataset for the LSTM neural network model. This model can predict the future update curve using inputs of various lengths and is capable of learning and predicting common update patterns of various client software modules. The final model chosen for deployment, which performed best in tests, has a 'dense' layer, outputs the number of requests, and an LSTM layer with 512 units. The network uses ReLU as the activation function and RMSprop and MSE as the error function.

The model was trained with early stopping to avoid overfitting. Each input batch had a length of 1440 (one day). The update curve prediction trained on data from one week from five servers had an MSE error of 0.00399 on the training set and an MSE error of 0.0040 on the test set. However, as seen in Figure 2, it failed to accurately model the curves.

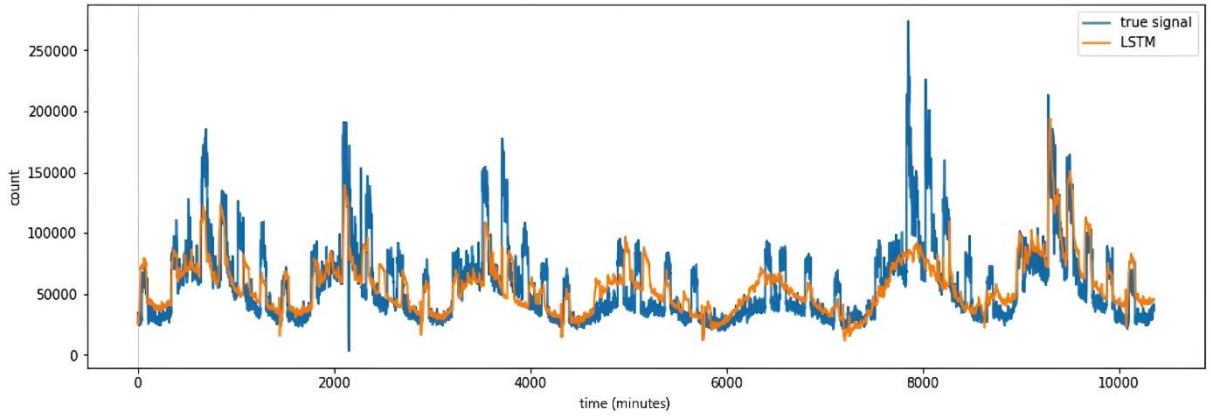

Figure 2: Prediction on 1 week train data.

Prediction on two weeks of data had a smaller MSE error of 0.0015 on the training set and the same error on the test set but still failed to accurately capture the curves.

Using fresh data containing 489,600 records collected over three months, the model achieved an MSE error of 0.0021 with a Mean Absolute Error (MAE) of 0.0305 on the test set. Predictions on a sample of 10,000 training records are shown in Figure 3 and on a sample of 10,000 test records in Figure 4. After two weeks of training, the error was larger, but the model was still able to achieve update curves, which was key for our upcoming anomaly classification.

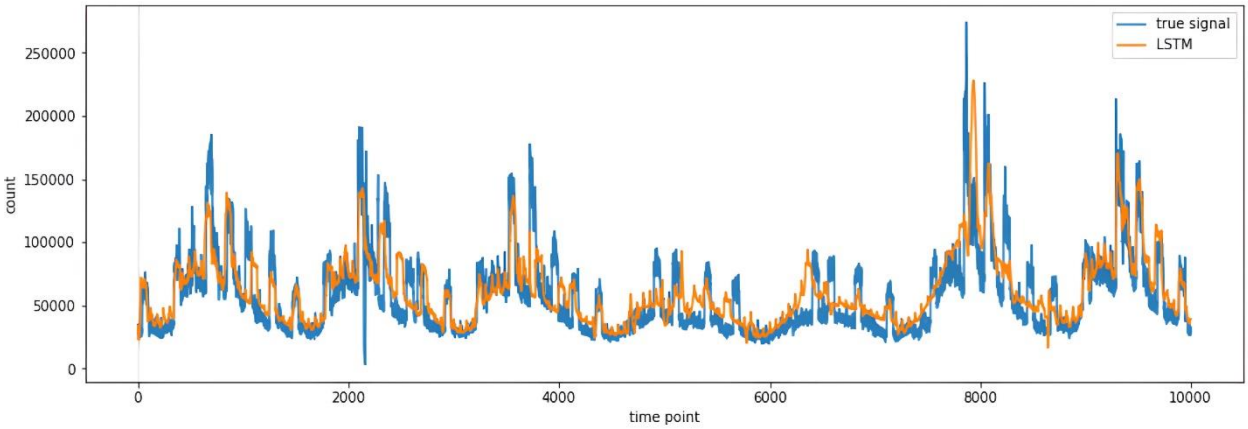

Figure 3: Prediction on train data on three months data.

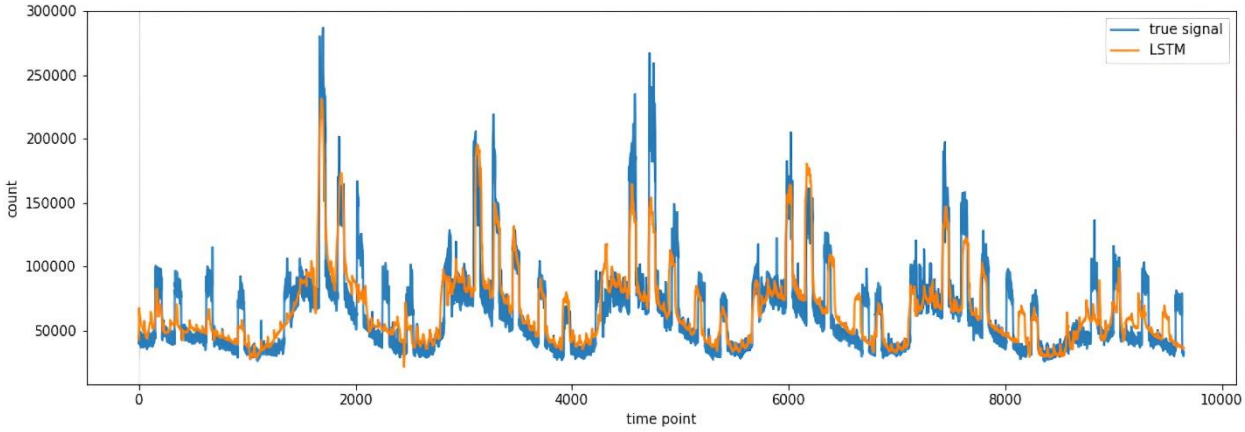

Figure 4: Prediction on test data on three months data.

The neural network output was used to identify various anomalies associated with updates. These anomalies were identified by comparing the anticipated update curve with the actual traffic and distinguishing them according to the following criteria:

1. Missing update: Anticipated load, usually lasting several hours, is missing.
2. Unusually large update: Because the load did not decrease, the update was not provided within one hour. Customers may not have received everything yet. The load continued for several hours.
3. Unexpected traffic: Client downloads began without our information, signaling heavier load. It could be New Year's or the day after holidays, someone might have just launched a new update.
4. Degraded performance: The volume of traffic is much smaller than expected.

We used a 5-minute window for classification. An increasing trend in the actual update curve, but a decreasing trend in the prediction, was used to characterize unexpected traffic. If the prediction indicated an increasing trend, but the actual update curve had a decreasing trend, it was considered a missing update. If the trends matched, a threshold based on the maximum value of the window was used to identify the last two groups.

Figure 5 shows the actual and expected curves in detail for one day. The system creates three alerts at the peak, which is where the selected time point is located. Since the actual curve is rising while the predicted curve is not, the curve was originally classified as an unexpected update. Then, as the number of actual update values was higher, it was classified as unexpected traffic, resulting in the classification of a missing update, because the number of actual update values was decreasing while the predictive curve was rising.

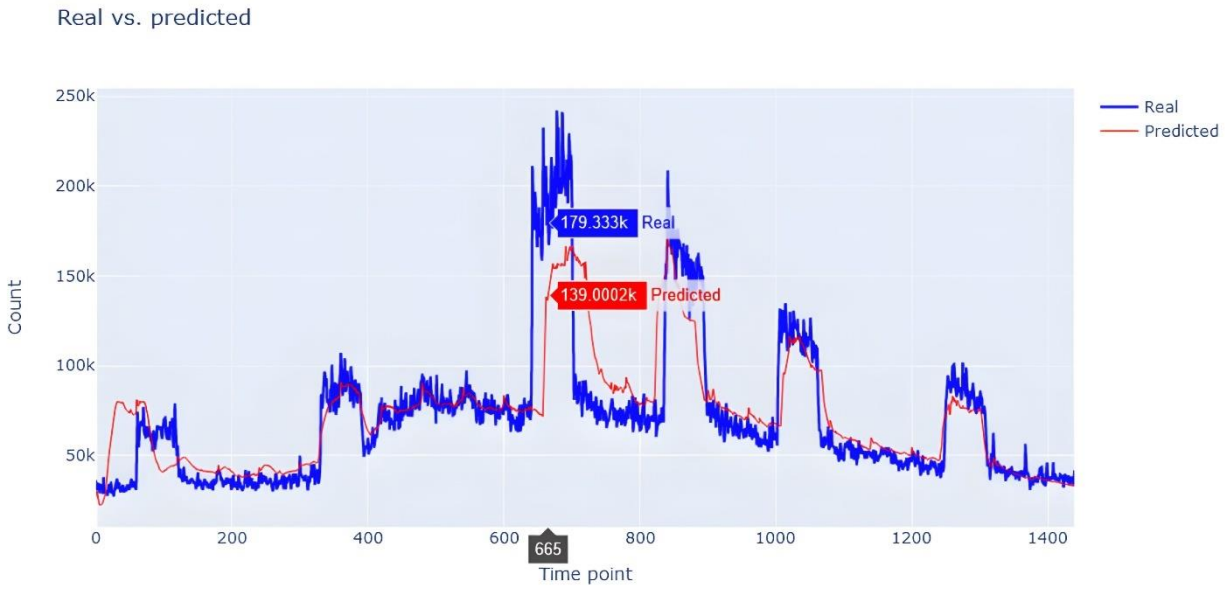

Figure 5: Real vs. one day ahead predicted traffic.

When creating alerts and real-time classification, we can change two factors: the threshold at which we decide to create an alert, or the categorization window. The classifier creates a significant number of alerts using a 5-minute window, providing good real-time response time with a threshold of 20,000.

When we increase the threshold value to 30,000, we create all the alerts that we physically see in Figure 5. We chose a threshold of 30,000, because increasing it further would prevent us from recognizing the first delayed update.

Results

The system was subjected to evaluation through two key steps. In the first part, we focused on the regression problem associated with load prediction. The second part dealt with the classification of anomalies based on predicted update curves.

Prediction Evaluation

The main tool of this phase was the LSTM neural network. Our data is unique as it relies on features from HTTP logs. Based on this specification, it was difficult to find comparable research. Vinayakumar et al. (Vinayakumar, 2017) reported that the predictive MSE value for their LSTM network was approximately 0.042. However, our value reached MSE 0.0025, suggesting better performance. We then analyzed several studies focused on similar topics. We compared our results with the outcomes of these studies and found that our models performed significantly better.

Further, we used the "Web Server Access Logs" dataset, which contains logs from the Iranian website zanbil.ir. Our model was modified to account for data from this dataset. After modification and training the model, we achieved an MSE value of 0.0225 on the test sample of our dataset and 0.035 on the test set of the Iranian website.

Expert Evaluation

The second phase of our system was real-time classification. Due to a lack of labeled data, we could not rely on standard evaluation, so we turned to expert analysis. Ing. Matej Březina, a specialist from ESET, helped us assess the usefulness of our model. Our model was able to identify missing updates and performance drops, which the expert confirmed. The real-time classifier generated alerts that matched the expert analysis.

Conclusion

We proposed a method for identifying and categorizing anomalies in HTTP logs that does not require in-depth expert analysis for removing anomalies from data in real time or labeling data, which requires professional and systemic knowledge. The effectiveness of the proposed detection system based on LSTM was evaluated in a real environment and proved successful in identifying various incursions during operation. Using a 5-minute window on predicted and actual update curves, we classified anomalies into four groups: Unexpected traffic (the load was high and we did not expect an update), unusually large update (we issued an update and the load was higher than usual), missing update (the load was low, but we did not expect an update) and decreased performance (the issued update led to lower load than usual).

Compared to existing solutions and expert analysis, the overall findings of the system and the performance of the neural network, which achieved an MSE value of 0.0021 on our noisy dataset, NRMSE and MSE values were several orders of magnitude better. The usefulness of the system was demonstrated by its ability to accurately predict update curves of noisy web server traffic even without automatic updates and can respond to events almost immediately, which is necessary to counter threats to the system and its infrastructure. An interesting aspect of this study is that it does not need a labeled or cleaned dataset and instead detects web server intrusions almost in real time using only HTTP logs of the web server.

In addition to comparing the performance of our model with the performance of other models trained on comparable data, we also used the model in a real context to demonstrate its applicability and conducted a performance analysis.

The contribution of this work lies in designing a technique that identifies and categorizes anomalies in HTTP logs without the need for deep expert analysis or data labeling, traditionally necessary for removing anomalies in real-time data. This innovative method is based on an intrusion detection system using LSTM, which proved effective in identifying various types of incursions in a real environment. The system accurately classifies anomalies into four groups in a 5-minute window for both, anticipated and current, update curves. Compared to existing solutions and expert analyses, the system demonstrates significantly better results with lower MSE values on noisy (real) data, proving its ability to accurately predict update curves and quickly respond to potential threats. The system operates effectively without the need for labeled or cleaned datasets and almost in real time detects network intrusions using only HTTP logs of the web server, highlighting its practical applicability and contribution in real-world scenarios.

Future Work

We will focus on eliminating thresholds and attempting more precise classification of update curve characteristics. For example, if an update appears earlier or later than expected, as shown in Figure 5, it may be advantageous to create just one alert by mapping certain combinations of alerts into a designated category. We cannot wait until the update curve drops, as the system must operate as close to real time as possible, so we will have to create more alerts if an anomaly occurs. Although aggregated alerts would be useful for a labeled dataset, the regression problem could be replaced by a classification problem. This way, the neural network could replace our window-based classifier, as it could classify an anomaly without the need for any further processing.

User Anomaly Detection

The dataset used in this study contains 15GB of compressed web server logs. These logs serve as a starting point for identifying potential user anomalies, such as targeted attacks or other suspicious behaviors. It is important to note that we do not know how many and what kind of anomalies are in the dataset. The goal of the study was to explore the dataset and obtain threshold values for attributes. The log format is described in Table 1.

The original dataset was processed by a parser, generating a list of users identified by their hardware identifiers and the time of their update requests. The General Data Protection Regulation (GDPR) prohibits the use of IP addresses. We also used the column with HTTP user agent information, which contains information about the client's system and language, as well as the number of individual HTTP requests, such as GET and HEAD, and server responses. A data sample that maps hardware identifier to specific web server access times is in Table 4. This dataset contains a table of the total number of user requests over a certain period of time. Each attribute name in Table 4 corresponds to a 30-minute interval. For example, attribute '3' represents the third 30-minute interval of the day, so the number in this property represents the number of requests in the 90th minute of the day. Since a day has 1,440 minutes, this represents 48 intervals of 30 minutes each.

Hardware_ fingerprint	get_ count	head_ count	country	system	status_200	...	status_304	0	...	47
XXXXXX...0	3	3	en_us	Windows	3	...	0	0	...	0.00
XXXXXX...1	2	2	es_es	Windows	0	...	2	0	...	0.00
XXXXXX...2	1	1	pl_pl	Windows	1	...	0	0	...	2
XXXXXX...3	5	425	en_us	Windows	10	...	420	0	...	12

Table 4: Sample from the dataset (hardware fingerprints were masked due to privacy).

It is important to emphasize that the dataset is not labeled, as user anomalies have not been studied on this dataset, so we do not know which sequences of requests to mark as anomalous. Consequently, we use an anomaly detection method that can detect these anomalies without the need for extensive expert evaluation and labeling. Looking at the graph of user request behavior in Figure 6, we see how diverse the behavior of just 10 random users is, and an expert would

need a lot of time to analyze the most anomalous behaviors even in such a small part of the dataset.

Figure 6: User request time plot.

The extracted dataset contains 8,484,251 unique users who made requests to a specific web server during one day.

Proposed Method

The techniques used in this study utilize Isolation Forests, which, unlike conventional techniques, are not based on density or distance for anomaly detection, saving time by not performing time-consuming calculations. This method had significant success in anomaly detection and is ideal for our large datasets obtained from web servers. Unlike clustering, which requires a lot of system memory, Isolation Forest can quickly find anomalies in our dataset.

First, we used DBSCAN to group users into a cluster and then analyzed outliers to identify common user behavior. However, this method was computationally inefficient, so we decided to use Isolation Forests, which do not attempt to model typical data and are therefore much more memory-efficient.

The two features of anomalous data are the basis of the Isolation Forest method. The amount of data as a whole is lower for anomalous data, and the attribute value of normal data differs greatly from that of anomalous data. In the training set, data are gradually separated using exclusively numerical values until the iTree can distinguish one data group from another.

Identification of Anomaly Thresholds

To identify threshold values for each property and better understand our data, we used the dataset shown in Table 4 for our first anomaly detection. Since data normalization only makes

sense when using linear approaches and does not affect the results of the Isolation Forest, we decided not to do it. Instead, we wanted to know the exact threshold values for each property, at which the model would no longer consider these values normal. We used all 8,484,251 records we collected, each representing one user, as well as 57 properties, such as the number of times each HTTP method was used, how many times each HTTP code was received, and how many update requests were made during all 30-minute daily intervals.

The "country" and "system" features were not used as input for the model, but rather for better understanding of the model. We used a contamination degree of "auto" because we did not know how many anomalies might be hidden in our data. For more accurate results, we used 100 estimators and set the "max features" parameter to the total number of properties.

Identification of Update Anomalies

We had to slightly modify the original dataset shown in Table 4 to detect unusual user activity. We want our model to show whether a certain sequence of requests is unusual or whether the ratio of requests created in a certain time window to all requests made by a specific user is outside the normal value.

The Isolation Forest method for this study used 48 properties, 100 estimators, and a contamination of 0.05, as we wanted to mark the top 0.5% of data as anomalous for further analysis.

Clustering

Clustering is an effective technique for analyzing large datasets and discovering patterns or groups in data. If we apply clustering to the results of the Isolation Forest, it can help in detecting and isolating anomalous data points. This method allows us to find and address outliers, which are often hidden in extensive datasets.

Data Point Selection

The performance of clustering depends on the selection of data points. To obtain a representative sample of anomalous data points, we first used the Isolation Forest technique to identify the 1,000 most anomalous users.

Scaling

Data needs to be scaled before clustering to avoid dominance of variables with a larger range. We used the StandardScaler for scaling data for its efficiency and simplicity.

Clustering Model

We chose the DBSCAN model for clustering anomalies identified using the Isolation Forest model based on our findings and the assumption that clusters will have a similar density.

DBSCAN

The behavior of DBSCAN is based on two main parameters - eps and min_pts. The values of these parameters affect the number and size of clusters. In our case, we set min_pts to twice the number of properties and used the "elbow" method to choose the optimal value of eps.

Visualization

A scatter plot was used to represent the dataset in Figure 7, where each cluster has a different color. The number of HEAD requests is represented on the Y-axis and the number of GET on the X-axis. The graph shows that users who sent the most GET and HEAD queries are grouped together.

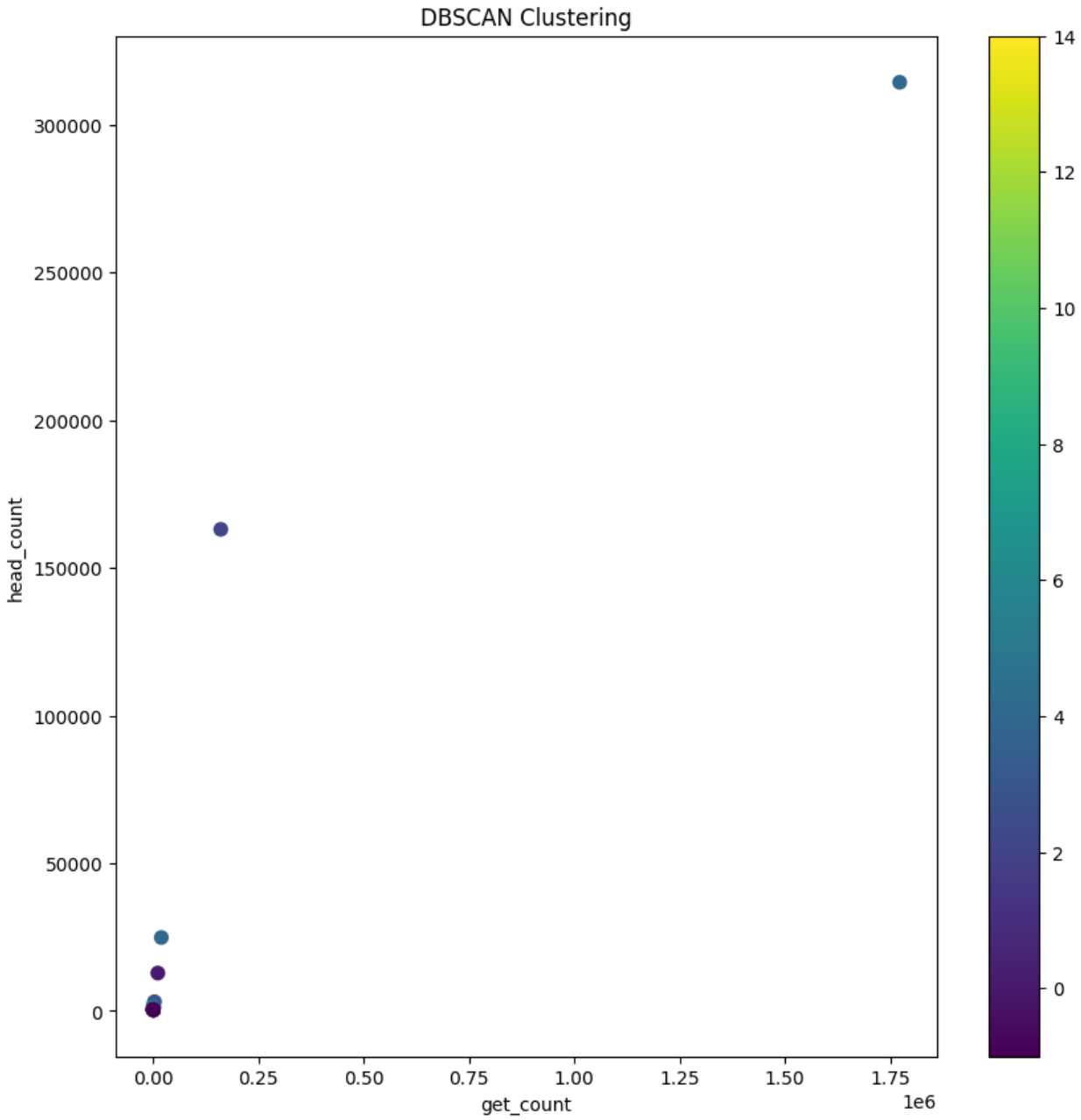

Figure 7: DBSCAN clustering: GET and HEAD count.

Detailed Analysis

To perform a thorough analysis, we had to reduce dimensionality. For this purpose, we used t-SNE. After running DBSCAN, we evaluated the results and visualized them using t-SNE and Plotly. Clusters created by DBSCAN show a linear pattern in the data, as you can see in Figure 7, where clusters correspond to a linear function where GET and HEAD requests are approximately the same. This may be considered unusual, as in normal online communication, GET requests are much more common than HEAD. Many clusters grouped users according to their respective countries.

Clustering of User Request Patterns

We clustered user requests based on the times they were made during the day. Instead of the exact number of user requests in a specific time period, we focused on the percentage value of his requests in this period. This was to identify unusual request patterns that may be shared by multiple users. To identify anomalous clusters, we used the elbow method. After repeatedly running DBSCAN with different values of min_pts and eps, we often obtained clusters with one or two users. With a value of 2 for the min_pts parameter, 991 points were identified as noise. In addition, we tested various values of the eps parameter for DBSCAN. In our case, the "elbow" point was at y = 8.5. Using an eps value of 8.5 and a min_pts value of 2, we obtained 978 points in cluster 0 and 21 points in cluster -1 (noise). We decided to send these 21 samples for further analysis. By clustering anomalous patterns of user requests, we were able to discover common patterns among different individuals classified as anomalous.

Expert Evaluation

To better understand the anomalous clusters discovered in our data, we sought the opinion of an expert in the field. The expert's views regarding user requests on the update server are as follows:

1. Regarding Figure 8, the expert stated:

"This looks normal, I assume these are users who are getting updates at a typical time. Overall, this graph looks healthy and as expected."

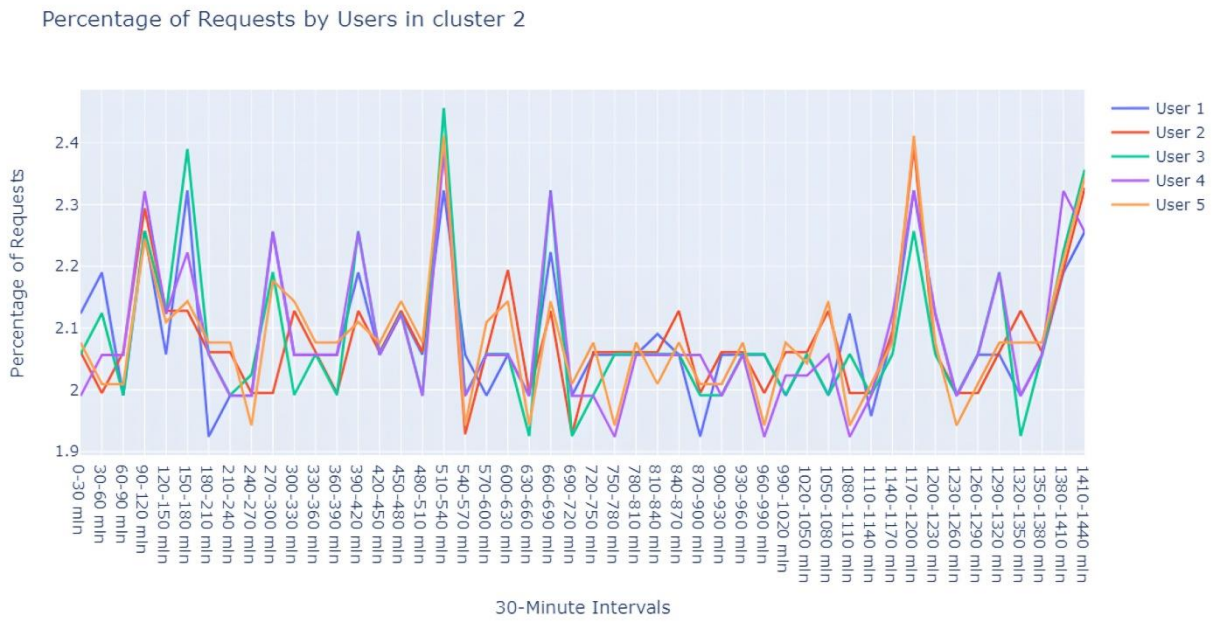

Figure 8: DBSCAN with 3 min_pts for cluster 2.

2. Regarding Figure 9, the expert stated:

"User1: Not sure. Maybe an external bot or mirror agent.

User2: These users may be logging in at a typical frequency, but they don't really benefit from it."

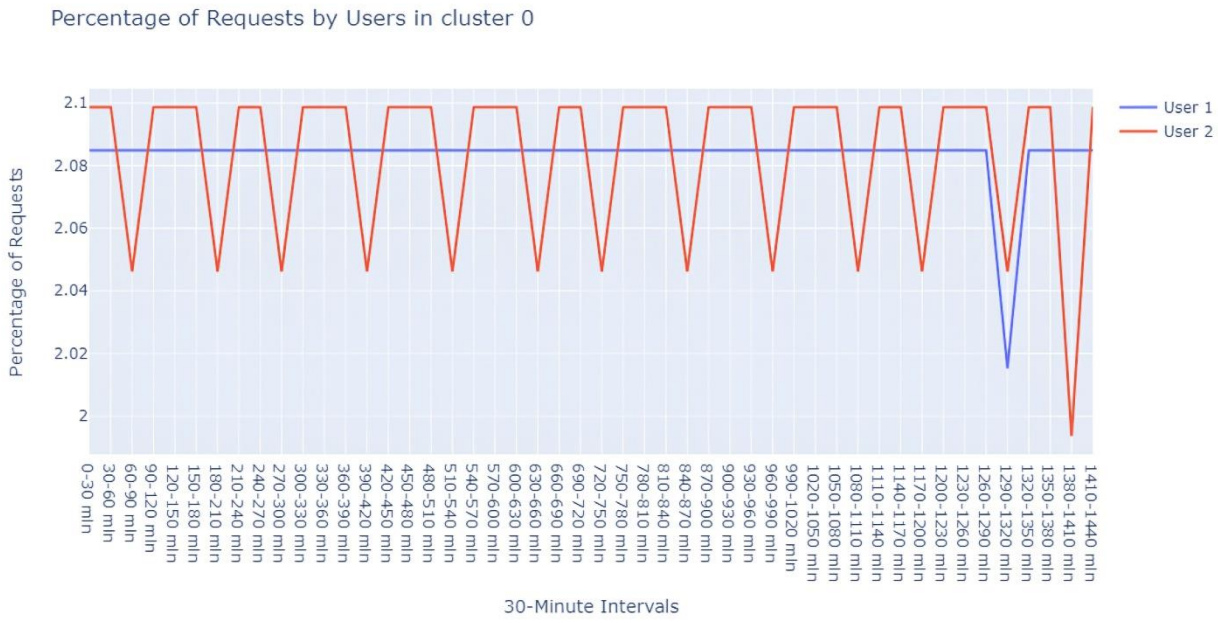

Figure 9: DBSCAN with 2 min_pts for cluster 0.

3. Regarding Figure 10, the expert stated:

"There is some correlation with update times. It seems that both users increase/decrease at the same time."

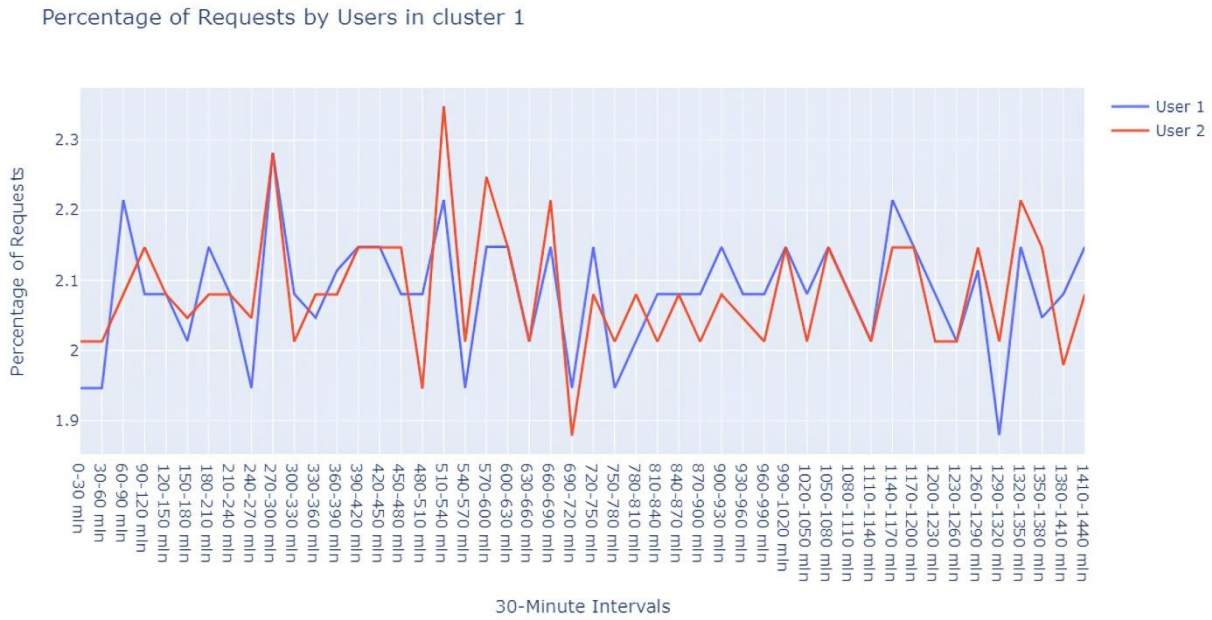

Figure 10: DBSCAN with 2 min_pts for cluster 1.

In conclusion, the expert's evaluation suggests that some user actions may be considered typical, while others may be unusual, such as the presence of possible external bots. There is a clear correlation between user behavior patterns and the times when updates are issued.

Results and Conclusion

By using the Isolation Forest method on our dataset, we were able to quickly identify anomalies among various users that we would normally overlook. We used two methods using Isolation Forest and both models were able to identify unique anomalies. The first method looked for broader anomalies in the dataset, while the second focused on anomalies in individual user behavior. This allowed us to better understand anomalous user behavior. If we want to analyze more data over a longer period, we will need to optimize CPU consumption.

The contribution of this research lies in an innovative approach to detecting anomalies in user behavior within a corporate network dataset, utilizing Isolation Forests. This methodology allows for accelerated identification of unusual activities, significantly reducing the need for extensive expert analysis and complex threshold setting, traditionally necessary for accurate detection. By implementing two different but complementary methods, both utilizing Isolation Forests, the study identifies not only basic anomalies in the dataset, but also performs detailed examination of individual user behaviors in requests, revealing specific anomalous patterns. The contribution of the research also provides security experts with an efficient tool for deep understanding and familiarization with the unique characteristics of their data and user behaviors. Rapid

identification and analysis of anomalous patterns with the proposed approach allows experts to gain insight into potential security risks and anomalies, supporting proactive security and enhancing their ability to predictively respond to potential threats.

Future Work

Future work will focus on creating a methodology for quick identification of user anomalies in real-time and better understanding of complex user behavior. This will allow us to identify patterns and create a labeled dataset for model accuracy evaluation. We can then test the model in infrastructure. We will also address the issue of different user time zones and aim to reduce CPU consumption of the model. In addition, we plan to conduct a detailed comparative study and compare our method with other studies.

Conclusion

We innovatively approached the prediction of update curves and later focused on automating anomaly detection among users with an emphasis on increased network security. The results highlight the significance of our techniques in enhancing real-time anomaly detection and protecting web servers.

Prediction of Update Curves

Our method effectively identifies anomalies in HTTP logs, eliminating the need for extensive expert analysis. Unlike many studies that are only applicable to specific datasets, our approach can be used with any HTTP log dataset. Testing in a real deployment confirmed its effectiveness.

The contribution of this study lies in introducing a new strategy that allows for identifying and categorizing anomalies in HTTP logs, without the need for in-depth analysis by experts or data labeling, which are usually required for real-time anomaly detection in data. The introduced technique, based on the LSTM system for intrusion detection, proves its effectiveness in recognizing various forms of intrusions in a real environment. This system accurately divides anomalies into four categories over a 5-minute interval, based on expected and actual update curves. Compared with existing solutions and expert analyses, this technique shows superiority, achieving lower MSE values on noisy data sets, thereby confirming its ability to accurately anticipate update curves and agilely respond to potential threats. It is important to note that the system effectively operates without labeled or modified data and identifies network disruptions almost in real time, using only HTTP logs of the web server, underscoring its practical applicability and contribution in real-world conditions.

Detection of User Anomalies

Automating the detection of outliers in large datasets provides network security analysts with faster responses to threats. The techniques we used accurately identify both general and specific anomalies in user behavior, offering a clearer understanding. The t-SNE method further helped visualize groups of users based on various attributes. Our analysis focused on user behavior and

their interactions with the server, using DBSCAN to identify anomalous patterns of user behavior, which are key for timely security interventions.

This study introduces a new technique that utilizes isolation forests for the rapid identification of anomalies in user behavior within a corporate network dataset. This new approach significantly reduces the need for intensive expert analysis and the complex process of setting threshold values, which are traditionally key for accurate anomaly detection. Through the application of two separate, yet complementary methods - both utilizing isolation forests - the research not only identifies general anomalies in the data set but also performs a detailed analysis of individual user requests and reveals unique anomalous patterns. Moreover, this innovative contribution provides security specialists with a practical tool that helps them understand and recognize the unique characteristics of their data and user behavior more deeply. With the ability to quickly identify and analyze anomalous patterns using the proposed approach, security experts can effectively gain insights into potential security risks and anomalies. This, in turn, supports the creation of a proactive security environment and enhances their ability to strategically preempt potential threats.

Reflection on Hypotheses

Reflection on Enhancing the Predictive Capability of Network Traffic Models for Web Servers in the Context of Intrusion Detection.

Our hypothesis assumed the benefits of improving network traffic predictions for better intrusion detection. Our research findings are:

1. Methodological contribution: We introduced a universal method for anomaly detection in HTTP logs, eliminating the need for expert intervention.
2. Testing in a real environment: The model effectively detected anomalies in a real network environment without a labeled dataset.
3. Utility of intrusion detection system: The LSTM-based system proved its usefulness in real-world conditions.
4. Anomaly classification: Our classification method enhanced understanding of network traffic and identified anomalies.
5. Performance metrics: The model outperformed other existing solutions in comparison.
6. Feature selection and new methodology: The study used a new approach utilizing only HTTP logs for intrusion detection without a pre-cleaned dataset. It used real, noisy data.
7. Evaluation in a real environment: Collaboration with an antivirus company practically confirmed our methodology.

Overall, our research supports the established hypothesis, presenting an effective solution for intrusion detection through understanding web server interactions.

Reflection on Anomaly Detection in User Behavior within a Web Server Environment for Intrusion Detection.

Our second goal was to explore modeling user behavior for anomaly detection. The hypothesis suggested recognizing anomalies through distinctive patterns of user behavior.

This study provides:

1. Approach to behavior modeling: Our method created a comprehensive user behavior model with a focus on automated anomaly detection.
2. Anomaly detection through behavioral changes: The techniques used detected anomalies, thereby confirming the hypothesis.
3. Use of machine learning: Our choice of the DBSCAN technique demonstrated the advantages of machine learning approaches.
4. Emphasis on automated network security analysis: Automation in our method allows for faster and more proactive responses to security threats.
5. Data visualization and comprehensive analysis: We expanded the hypothesis using the t-SNE technique for deeper data analysis.
6. Broad applicability: The flexibility of our technique confirms its wide applicability.

By summarizing the methods and conclusions, we confirm our original hypothesis, positioning user behavior modeling as a key component of network security.

STU

List of articles and publications

Published articles

1. Benova, Lenka, and Ladislav Hudec. "Web server load prediction and anomaly detection from hypertext transfer protocol logs." International Journal of Electrical & Computer Engineering (2088-8708) 13.5 (2023).

   Citations:

   Golis, Tomáš, Pavle Dakić, and Valentino Vranić. "Automatic Deployment to Kubernetes Cluster by Applying a New Learning Tool and Learning Processes." Proceedings http://ceur-ws. org ISSN 1613 (2023): 0073.

2. Benova, Lenka, and Ladislav Hudec. "Using Web Server Logs to Identify and Comprehend Anomalous User Activity." 2023 17th International Conference on Telecommunications (ConTEL). IEEE, 2023.

3. Benova, Lenka, and Ladislav Hudec. "Detecting anomalous user behavior from NGINX web server logs." 2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC). IEEE, 2022.

   Citations:

   Golis, Tomáš, Pavle Dakić, and Valentino Vranić. "Automatic Deployment to Kubernetes Cluster by Applying a New Learning Tool and Learning Processes." Proceedings http://ceur-ws. org ISSN 1613 (2023): 0073.

4. Smolen, Timotej, and Lenka Benova. "Comparing Autoencoder and Isolation Forest in Network Anomaly Detection." 2023 33rd Conference of Open Innovations Association (FRUCT). IEEE, 2023.

   Citations:

   Golis, Tomáš, Pavle Dakić, and Valentino Vranić. "Automatic Deployment to Kubernetes Cluster by Applying a New Learning Tool and Learning Processes." Proceedings http://ceur-ws. org ISSN 1613 (2023): 0073.

Under review articles

Benova, Lenka, and Ladislav Hudec. "Comprehensive Analysis and Evaluation of Anomalous User Activity in Web Server Logs." Sensors, under review (submitted on 2023-12-19).

STU

Bibliography

Adari., S. A. (2019). Beginning anomaly detection using python-based deep learning. *Springer*.

Bertero, C. a. (2017). Experience report: Log mining using natural language processing and application to anomaly detection. *2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 351-360.

Casado-Vara, R. a.-P.-l.-F.-V. (2021). Web Traffic Time Series Forecasting Using LSTM Neural Networks with Distributed Asynchronous Training. *Multidisciplinary Digital Publishing Institute*.

Debnath, B. a. (2018). LogLens: A real-time log analysis system. *2018 IEEE 38th international conference on distributed computing systems (ICDCS)*, pp. 1052-1062.

Du, M. a. (2017). Deeplog: Anomaly detection and diagnosis from system logs through deep learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285-1298.

Gao, Y. a. (2017). Anomaly detection of malicious users' behaviors for web applications based on web logs. *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pp. 1352-1355.

Huo, Y. a. (2019). Long-term span traffic prediction model based on STL decomposition and LSTM. *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), IEEE*, pp. 1-4.

Iwan Syarif, A. P.-B. (2012). Unsupervised clustering approach for network anomaly detection. *International conference on networked digital technologies, Springer*, pp. 135-145.

Kim, T.-Y. a.-B. (2018). Web traffic anomaly detection using C-LSTM neural networks. *Expert Systems with Applications, Elsevier*, pp. 66-76.

Malaiya, R. K. (2018). An empirical evaluation of deep learning for network anomaly detection. *2018 International Conference on Computing, Networking and Communications (ICNC)*, pp. 893-898.

Monowar H Bhuyan, D. K. (2017). Network traffic anomaly detection and prevention: concepts, techniques, and tools. *Springer*.

Naseer, S. a. (2018). Enhanced network anomaly detection based on deep neural networks. *IEEE access*, pp. 48231-48246.

Naveed Chouhan, A. K. (2019). Network anomaly detection using channel boosted and residual learning based deep convolutional neural network. *Applied Soft Computing, p. 105612*.

Ni, X. (2016). Network anomaly detection using unsupervised feature selection and density peak clustering. *International Conference on Applied Cryptography and Network Security, Springer*, pp. 212-227.

Radford, B. J. (2018). Network traffic anomaly detection using recurrent neural networks.

Shelatkar, T. a. (2020). Web traffic time series forecasting using ARIMA and LSTM RNN. *ITM Web of Conferences, EDP Sciences*.

Torres, P. a. (2017). Data analytics for forecasting cell congestion on LTE networks. *2017 Network Traffic Measurement and Analysis Conference (TMA)*, pp. 1-6.

Vartouni, A. M. (2018). An anomaly detection method to detect web attacks using stacked auto-encoder. *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), IEEE*, pp. 131-134.

Vinayakumar, R. a. (2017). *Applying deep learning approaches for network traffic prediction.* 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE.

Wu, N. a. (2020). Deep transformer models for time series forecasting: The influenza prevalence case.

Xu, J. a. (2021). Anomaly Transformer: Time Series Anomaly Detection with Association Discrepancy.

Zhao, N. a. (2021). An empirical investigation of practical log anomaly detection for online service systems. *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 1404-1415.

Zhao, W. a. (2021). Network Traffic Prediction in Network Security Based on EMD and LSTM. *Proceedings of the 9th International Conference on Computer Engineering and Networks, Springer*, pp. 509-518.

Zhao, Y. a. (2018). Forecasting wavelet transformed time series with attentive neural networks. *2018 IEEE International Conference on Data Mining (ICDM), IEEE*, pp. 1452-1457.

Zhou, H. a. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of AAAI*.