3RD WORKSHOP ON MACHINE LEARNING IN NETWORKING (MaLeNe) PROCEEDINGS

SEPTEMBER 1, 2025

CO-LOCATED WITH
THE 6TH INTERNATIONAL CONFERENCE ON NETWORKED SYSTEMS (NETSYS 2025)
ILMENAU, GERMANY

Client–Agnostic Continuous Authentication via Keystroke–Induced Traffic Patterns

Alexander Niedermayer, David Monschein, Oliver P. Waldhorst
Institute of Data–Centric Software Systems (IDSS), Karlsruhe University of Applied Sciences
{nial1016, david.monschein, oliver.waldhorst}@h-ka.de

***Abstract*—How can we continuously verify the identity of users without modifying their devices? We introduce a client–agnostic method that leverages keystroke–induced network traffic patterns to passively authenticate users. It can be deployed on infrastructure already common in network environments. By applying contrastive learning to Web–Socket packet traces, we compare new traffic against previously seen patterns from the same user. In an experiment with 75 users, our method achieved 87 % accuracy–improving over a statistical baseline by 27.7 percentage points (pp). These results demonstrate that network traffic captures meaningful behavioral signatures and can serve as a foundation for practical, continuous user authentication.**

***Index Terms*—Continuous Authentication, Network Monitoring, Network Security.**

I. Introduction

Nowadays, many people use web–systems ranging from streaming services, over online games, to highly sensitive online banking applications. Even though authentication is such an important aspect, many applications solely rely on knowledge-based approaches like passwords. These have been found susceptible to different kinds of attack vectors. Some examples are dictionary attacks and heat analysis [14].

The impact of these attacks can be reduced by continuously authenticating the user's identity. An easy approach would be to request the user's credentials in short intervals. While being effective, this strongly decreases the usability of the application. Based on this issue, the field of behavioral authentication tries to offer a solution and provide an additional layer of security without sacrificing usability.

Recent behavioral authentication approaches are based on metrics derived from the user interacting with the device [20]. These have been proven to be an effective way of matching users to their behavior, but rely heavily on client–sided modifications [22, 26]. These modifications are needed to capture keyboard or touch–screen inputs. This can be an obstacle for two main reasons. First, the client needs to be modified to collect the necessary user behavior. Second, the collection and sending of the user behavior might have negative impact on the battery life of the device and use additional mobile data.

We propose a new approach, relying on the continuous authentication of users based on their network traffic generated by using an application. The system takes a time series of the exchanged packets as input for a machine learning model. This eliminates the need for client–sided modifications and thus does not have any impact on battery life or used mobile data. Our approach can be injected inside an existing system with minimal changes, as seen in Fig. 1. The contributions of this paper are, therefore:

- An approach for continuous authentication without application changes on the client side.
- An evaluation of the proposed approach using a dataset, generated from keyboard inputs.
- Empirical evidence that network traffic captures non–trivial human behavior and can be used for continuous authentication.

We evaluated our approach in an experiment with 75 users. The results show an improvement in accuracy of 27.7 pp compared to a statistical baseline approach and 37 pp compared to a random guess. Another finding is the strong association between the user's typing behavior and the generated web–socket packets, which further supports the claim of using network traffic for user authentication.

This paper is structured as follows. In Section II we provide an overview of the foundation and related work in the field of behavioral authentication and network traffic analysis. Section III describes our approach for continuous authentication based on network traffic. The experiment setup and results are presented in Section IV-B. Finally, we discuss possible limitations of our approach in Section V and conclude our work in Section VI.

II. Related Work

In this section, we will give an overview of the two main fields, that are the foundation for our work. The first one is a brief summary of behavioral authentication. The second one covers the basics of network traffic analysis using machine learning approaches.

A. Behavioral Authentication

Behavioral authentication offers a supporting way of authenticating users. The main goal is, to make knowledge-based approaches more secure, by collecting and analyzing user behavior. Currently a lot of focus lies on the analysis of the users physical interaction with the device.

The device interaction can be captured in different ways [14]. One of the most common ways is to capture the users typing behavior. This approach has been shown by multiple publications [1, 4], to be a reliable method of using the user behavior for additional authentication information.

The rise of the Internet of Things (IoT) devices has also brought up new methods of capturing user behavior. For

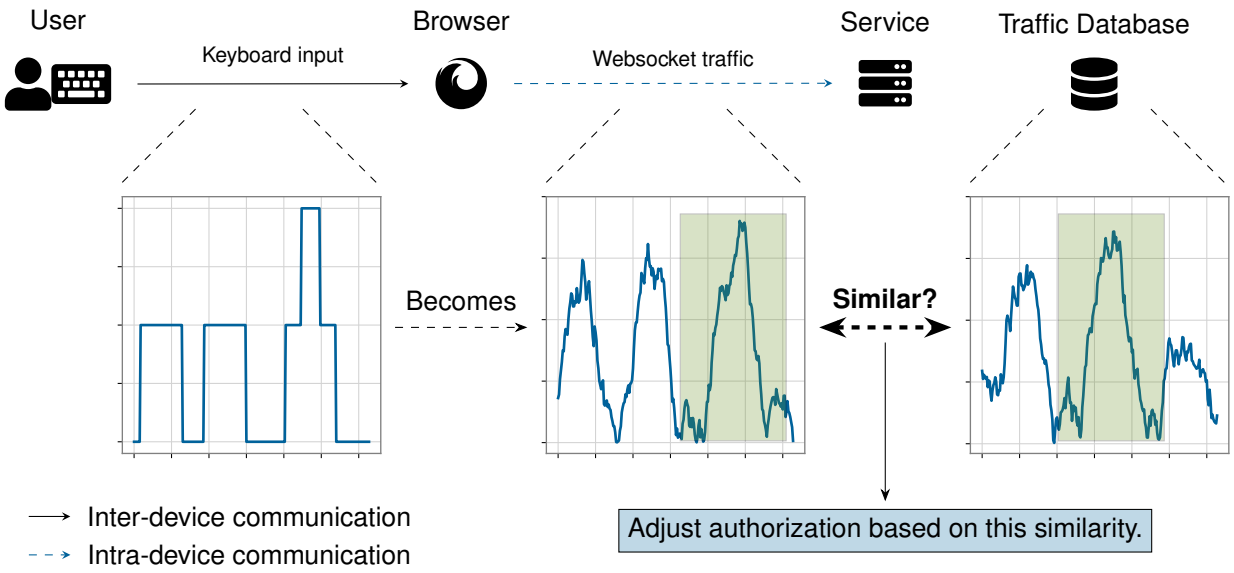

Fig. 1. Overview of our network–traffic–based continuous authentication approach.

example, a smartphone offers many different sensors like touchscreens, accelerometers, gyroscopes and more [3]. Additionally smartwatches even offer the possibility of capturing the user's heartbeat, breathing rhythm or other health related data [14]. These behavior metrics have also been shown, to be usable for user authentication [12].

Another approach would be, to combine user behavior and device attributes, by correlating user behavior to internal device timing characteristics. This has been done in [19]. The key insight is, that when users interact with a device—through typing, mouse movements, or gestures—the resulting input events are not only shaped by the user's behavior but also by the device's internal processing and sampling mechanisms. Specifically, many human-interface devices (like keyboards and touchpads) sample inputs at fixed intervals determined by hardware clocks. These clocks introduce subtle, device-specific timing patterns into the recorded input stream. When analyzing the timing of these events, the fixed sampling rates can create distinct frequency peaks in the input data, which then can be used to identify the device and, by extension, the user. In the mentioned work, the authors demonstrate a rank–1 accuracy of 84.6 % for 10,000 devices when using this approach.

B. Network Traffic Analysis with Machine Learning

Currently, the most common use–case for network traffic analysis is the detection of anomalies and attacks [2]. This is often done by implementing so called Intrusion Detection Systems (IDS). A frequent use–case for machine learning in the field of IDS, is the recognition of Distributed Denial of Service (DDoS) attacks [11, 5]. Overall the scope of IDS extends far beyond DDoS attacks and can also focus on common exploits and malware [10, 29].

For this task different algorithms ranging from simple rule–based approaches to deep learning models have been used [6]. In recent deep learning publications, contrastive learning has shown to be a promising approach for network traffic analysis. Although results exceeded 90 % balanced accuracy, the classification tasks were limited to binary classification (normal/malicious traffic) [13, 16, 28] or application identification (video streaming, file download...) [17].

C. Contrastive Learning

The main idea behind contrastive learning is to learn a vector representation of the input data, that has a short distance for similar inputs and a long distance for dissimilar inputs [9, 27]. A loss function for this optimization problem can be defined as:

$$\mathcal{L} = \sum_{i \in I} \mathcal{L}_i = -\log \left(\frac{e^{z_i \cdot z_{j(i)}}}{\sum_{k \in S(i)} e^{z_i \cdot z_k}} \right) \tag{1}$$

With x_i being an example input and $z_i = f(x_i)$, the function $j(i)$ delivering a index where $x_{j(i)}$ is from the same class as x_i. The function $K(i)$ delivers a set of indices, where $x_{k \in K(i)}$ is from a different class than x_i. Furthermore $S(i) \subseteq K(i)$. There exist different strategies to sample $S(i)$, for example random sampling or hard negative sampling [8].

This formulation makes contrastive learning particularly well–suited for our user recognition task, as it does not require a fixed number of classes. Unlike traditional n–class classification, contrastive learning operates on similarity between samples, allowing the model to scale to an arbitrary number of users.

III. Methodology

The following section describes the methodology of our continuous authentication system. We start by introducing the main idea behind our approach. Afterwards, we describe the implementation of our system.

A. Main Idea

The main idea is, that during event–based communications between client and server, the characteristics of user interaction are causally connected to the network traffic. This offers the possibility for behavior being represented in network metrics, especially packet frequency, packet inter–arrival time and packet size.

For example, consider a web application with a typing input. If every typing input triggers a packet exchange, the packet frequency can be correlated to the typing speed. Also different patterns, when typing key combinations might be represented in the data stream. Stragapede et al. [25] have already successfully identified users based on typing events, so the transfer to causally connected network events is a logical step.

B. User Recognition with Web–Socket Packets

The architecture of our approach is shown in Fig. 2. As input, we receive a list of web–socket packets that were sent between the client and the server. The web–socket packets are timestamped on arrival and contain the payload of the message. We calculate the difference between the receive time of two consecutive packets, giving us the packet inter–arrival time as first time series. The second time series in our approach is the effective change a packet has on the text input. Having access to the payload of the message, we can extract, whether the packet adds a sequence of characters to the text input or deletes a sequence of characters. Both of these time series are normalized, using the minimum and maximum value, derived from the respective training data.

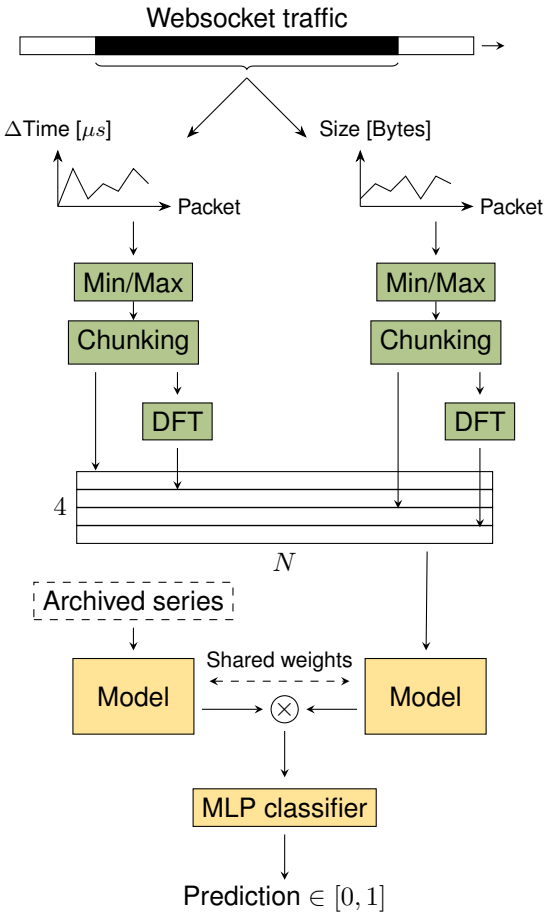

Fig. 2. Architecture of the user recognition on the packet level.

For better processing, the resulting time series are split into chunks of a fixed size $N = 2^k \in \mathbb{N}$. For each chunk we also calculate the discrete fourier transform (DFT) of the packet inter–arrival times and the number of character changes per package. This leaves us with a final 2–dimensional array with the shape $N \times 4$.

For the contrastive learning method, these chunks are then combined into matching and non–matching pairs. Every chunk for a user is randomly matched with another chunk of the same user. The same is done for non–matching pairs. Every chunk of a user is randomly matched with a chunk of another user. A dataset with $S \in \mathbb{N}$ samples will then produce S matching and S non-matching pairs, resulting in a dataset with $2S$ samples.

Our system is based on a siamese neural network architecture [18]. Siamese networks are composed of two identical sub–networks, that share the same parameters and weights. They are used for comparing two inputs. Each input is being forwarded through a sub–network, that extracts important features, represented in the embedding space. Afterwards, a distance metric is being applied to the embeddings, which will output a similarity score.

Since a multivariate time series can be seen as a grayscale image, we take a similar approach to image recognition tasks. We use a deep learning model as backbone, that is trained to generate an embedding of the input data. This is done for each of the two input time series. The embeddings are then multiplied and the result is passed through a multi–layer perceptron (MLP), that outputs the similarity score of the two input time series.

IV. Evaluation

This section starts by presenting the experimental setup, developed for creating a dataset of network traffic, generated by user inputs. We then evaluate our method from the previous section, using this dataset. We also compare our approach to a baseline method, as evidence for substantial user behavior being captured in the network traffic.

A. Generation of User–behavior driven Network Traffic

To generate a dataset of network traffic, we are using the popular web–service *Overleaf*[1]. Overleaf is an online LaTeX editor that allows users to collaborate on documents in real–time. The service uses a web–socket connection, to send updates to the server, whenever a user interacts with the document. This results in network traffic that is heavily influenced by the users typing behavior.

[1]https://github.com/overleaf/overleaf

For capturing the exchanged network traffic, we use the open–source tool *mitmproxy*[2]. Mitmproxy is modified, to store every forwarded web–socket packet in a log file. The log file contains the timestamp of the packet and the respective payload. The payload of the packet is then used to extract the transmitted character changes in the text. Furthermore, we also remove the acknowledgements (ACKs) from the packet stream, since they do not contain any user behavior information.

The dataset is generated, by using recorded keystrokes from the *Keystroke Verification Challenge* dataset [23]. We take every user data that spans over a time period of 15 minutes for desktop and one hour for mobile devices. Afterwards, a script is used, to control the browser and simulate the typing behavior of the given user. The script inputs the keystrokes into the Overleaf editor, which then generates the network traffic. The keystrokes are replayed in real–time, to get an accurate representation of the corresponding network traffic. An overview of the experiment can be seen in Fig. 3.

For our evaluation, we are using a ResNet18 [7] model that is trained on the generated dataset. The dataset consists of traffic generated from 75 different users. The dataset is balanced, so that every user maps to the same amount of traffic. This leaves us with a total time of 18 hours and 45 minutes of recorded network traffic.

The optimization problem is solved using the AdamW [15] optimizer with the standard learning rate of 0.001 and standard weight decay of 0.01. The model is trained for 100 epochs with a batch size of 256. Our loss function for the classification task is the binary cross–entropy loss.

B. Results

We use a 5–fold cross–validation to evaluate the model with specific dataset splits. Metrics are always given as mean over the best score from the 5–folds with a standard deviation. We select the best score as epoch with the highest accuracy. Precision, Recall and F1–Score are also given from this epoch.

As additional metric, we provide a baseline using a decision tree classifier. The classifier is trained using the mean, standard deviation, maximum and minimum value of each feature time series. The evaluation for the decision tree is done using the same 5–fold cross–validation.

[2]https://mitmproxy.org/

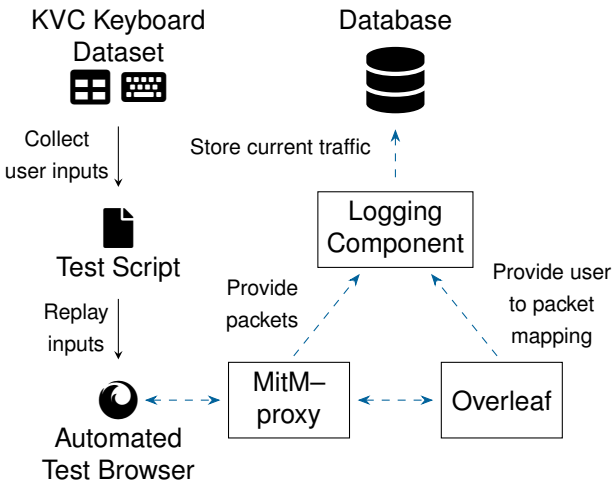

Fig. 3. Overview of our dataset generation setup we used for evaluating our continuous authentication approach.

In Table I we present the results of our evaluation for different chunk sizes N. With a chunk size of 64 records, we achieve an average accuracy of 87 % with a standard deviation of 1 pp. Furthermore, the results show, as expected, that a longer observation time results in a higher accuracy. For both models, we observed, that this difference can lie in the range of up to 11 pp for classification accuracy, when going from a chunk size of 16 to 64. The area under the curve (AUC) of the receiver operating characteristics (ROC) curve gives a value of 93.19 % $\pm$ 1.74%. This high score should be taken with a grain of salt, since as seen in Table I, the measured precision value is at 82.5 %. The ROC curve is displayed in Fig. 4, representing the average performance over a 5–fold cross-validation. The shaded area around the curve indicates the standard deviation, providing a measure of variability across the folds.

Figure 5 evaluates the impact of different preprocessing methods on the accuracy of our approach. The x–axis represents different chunk sizes N and the y–axis shows the classification accuracy in percent. The best results were achieved, when having full access to the packet body. Removing the

TABLE I
METRICS OF THE USER RECOGNITION BASED ON NETWORK TRAFFIC. RELATIVE TO CHUNK SIZE N.

Metric	Model	$N = 16$	$N = 32$	$N = 64$
Accuracy	Our approach	75.9 % $\pm$ 4.3 %	82.8 % $\pm$ 3.0 %	**87.00 %** $\pm$ 1.0 %
	Baseline	61.8 % $\pm$ 7.3 %	64.1 % $\pm$ 6.8 %	59.3 % $\pm$ 3.6 %
Precision	Our approach	71.2 % $\pm$ 2.4 %	83.2 % $\pm$ 5.6 %	**82.5 %** $\pm$ 1.8 %
	Baseline	58.9 % $\pm$ 5.5 %	61.6 % $\pm$ 5.7 %	58.1 % $\pm$ 4.9 %
Recall	Our approach	87.1 % $\pm$ 5.3 %	83.0 % $\pm$ 8.2 %	**94.0 %** $\pm$ 2.0 %
	Baseline	79.7 % $\pm$ 10.9 %	80.8 % $\pm$ 8.1 %	76.0 % $\pm$ 12.0 %
F1–Score	Our approach	78.3 % $\pm$ 1.6 %	82.6 % $\pm$ 3.1 %	**87.85 %** $\pm$ 0.8 %
	Baseline	67.4 % $\pm$ 6.3 %	69.4 % $\pm$ 3.3 %	64.9 % $\pm$ 1.8 %

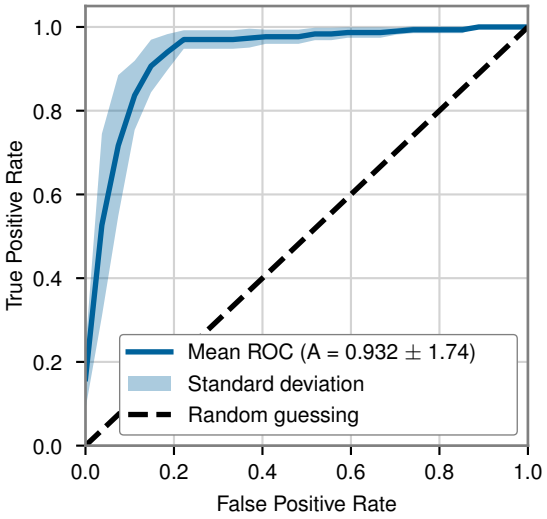

Fig. 4. Average ROC curve derived from the folds of our siamese–network approach with $N = 64$.

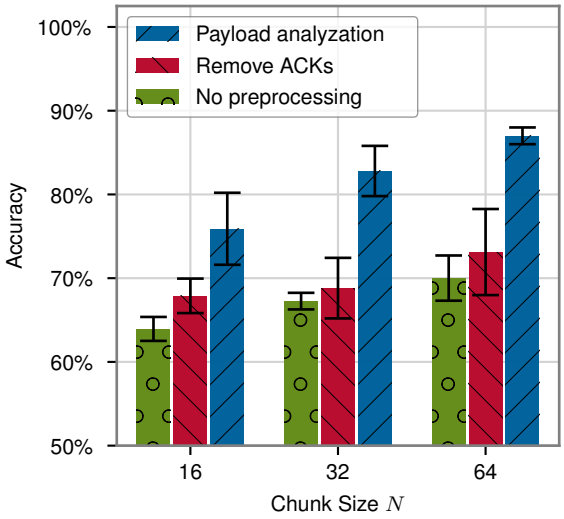

Fig. 5. Accuracy comparison of the siamese network using different preprocessing methods for different chunk sizes N.

ACKs from the packet stream yields a small improvement compared to working on the raw packet stream. The accuracy improvement lies in the range of around 3 pp for $N \in \{16, 64\}$ and 1.5 pp for $N = 32$. The analysis of the character changes transmitted in the packet body and removal of ACKs shows an increased accuracy of 14 pp for $N \in \{32, 64\}$ and 8 pp for $N = 16$, when comparing to just removing the ACKs.

V. Discussion

Compared to the baseline, our approach shows an improvement in the range of 14 pp to 27 pp in accuracy. We hereby argue that due to the improvement of our approach over the baseline, we are able to capture more user behavior inside the network traffic, than just the average typing speed.

We come to this conclusion, since the baseline model is trained on the average, standard deviation, maximum and minimum value of the two time series. The average value of the character changes per packet and the average typing speed of the user are correlated. This is also shown in Fig. 6, where the correlation coefficient $r = 0.78$ indicates a strong association between these two variables. This association is also causally linked, since more keyboard inputs lead to more transmitted characters. The average typing speed of a user should therefore be present in the average transmitted characters. Going further, this metric is also available to the baseline model. Due to the improvement of our approach over the baseline, there must be more user behavior captured in the network traffic, than just the average typing speed.

Nonetheless, when compared to methods that rely on direct keystroke data, there is still room for improvement. Recent publications on the same dataset yield results of around 92 % accuracy [24]. Therefore there might be a loss of information, when transferring the keystrokes to network traffic.

Besides our contributions there are also two main limitations in our approach. First, our current method needs access to the packet body, to extract the required features. This is not always possible, for example when using Transport Layer Security (TLS). A common solution would be to employ a TLS proxy, which decrypts the traffic for inspection before re-encrypting it and forwarding it to the destination. The latter is a common practice in many organizations, with little to no impact on the user experience [21].

The second limitation of our approach is that it currently requires a time window of approximately 15 minutes for effective analysis. This constraint arises from the lower frequency of network traffic events compared to direct keyboard inputs, which limits the granularity and immediacy of the captured behavioral signals. Reducing the required observation time

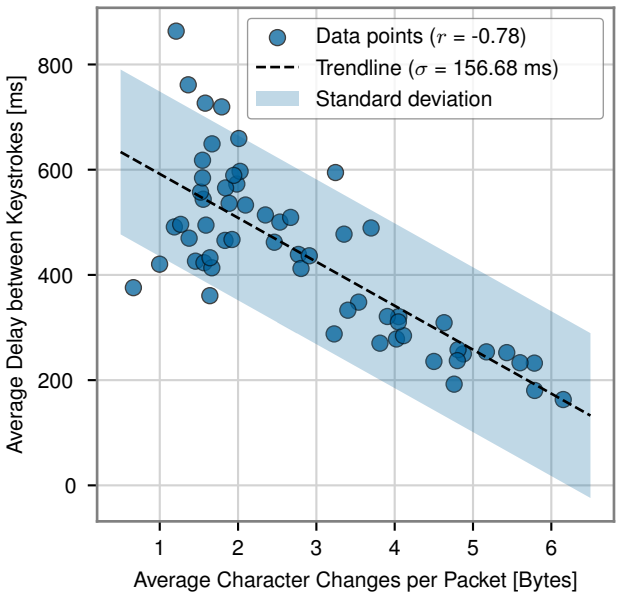

Fig. 6. Correlation between the average delay between keystrokes from the KVC dataset [23] and the average character changes per packet from our dataset.

could significantly enhance the practicality of the method, especially for real-time applications. This may be achievable with access to a larger and more diverse dataset containing longer and more varied user sessions, which would enable better generalization and potentially allow the model to detect patterns in shorter time spans.

VI. Conclusion

Our work showed, that user recognition based on network traffic can be possible. While achieving good results of 87 % accuracy, methods using the direct keystroke data still yield better results. Additionally our work offers evidence, that more user behavior is captured in the network traffic, than just the average transmitted characters. Therefore user–centric network traffic analysis is a promising approach for continuous user authentication, without the need for client–sided modifications.

In respect of future work, improvements regarding the machine learning setup are possible. Our proof of concept used a siamese–network with a convolutional network. While achieving good results on comparison to the needed computing power, we are still curious about the potential of more complex approaches. This might improve the accuracy and may also reduce the observation window.

At last there might also be the possibility to achieve similar results with a more secure and privacy–preserving approach. Maybe there is enough information hidden in TLS encrypted network traffic, that can be used for user recognition. Another common sight in large–scale network is the use of so called aggregated flows for network traffic analysis. These flows are a summary of the network traffic and can be used for anomaly detection. Recognizing users based on these flows might be another privacy–centric approach for continuous user authentication.

Code and Data Availability

The repository https://doi.org/10.5281/zenodo.15579140 contains the code and data used for this work.

Acknowledgments

This work was supported by the project bwNET2.0 funded by the Ministry of Science, Research and the Arts Baden–Württemberg (MWK).

References

[1] Alejandro Acien et al. “TypeNet: Deep learning keystroke biometrics”. In: *IEEE Transactions on Biometrics, Behavior, and Identity Science* 4.1 (2021), pp. 57–70.

[2] Ijaz Ahmad et al. “Machine learning meets communication networks: Current trends and future challenges”. In: *IEEE access* 8 (2020), pp. 223418–223460.

[3] Abdulaziz Alzubaidi and Jugal Kalita. “Authentication of smartphone users using behavioral biometrics”. In: *IEEE Communications Surveys & Tutorials* 18.3 (2016), pp. 1998–2026.

[4] Salil P Banerjee and Damon L Woodard. “Biometric authentication and identification using keystroke dynamics: A survey”. In: *Journal of Pattern recognition research* 7.1 (2012), pp. 116–139.

[5] Narmeen Zakaria Bawany, Jawwad A Shamsi, and Khaled Salah. “DDoS attack detection and mitigation using SDN: methods, practices, and solutions”. In: *Arabian Journal for Science and Engineering* 42 (2017), pp. 425–441.

[6] Anna L Buczak and Erhan Guven. “A survey of data mining and machine learning methods for cyber security intrusion detection”. In: *IEEE Communications surveys & tutorials* 18.2 (2015), pp. 1153–1176.

[7] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[8] Yannis Kalantidis et al. “Hard negative mixing for contrastive learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 21798–21809.

[9] Prannay Khosla et al. “Supervised contrastive learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 18661–18673.

[10] Ansam Khraisat et al. “Survey of intrusion detection systems: techniques, datasets and challenges”. In: *Cybersecurity* 2.1 (2019), pp. 1–22.

[11] Samuel Kopmann, Hauke Heseding, and Martina Zitterbart. “HollywooDDoS: Detecting Volumetric Attacks in Moving Images of Network Traffic”. In: *2022 IEEE 47th Conference on Local Computer Networks (LCN)*. IEEE. 2022, pp. 90–97.

[12] Antwane Lewis, Yanyan Li, and Mengjun Xie. “Real time motion-based authentication for smartwatch”. In: *2016 IEEE Conference on Communications and Network Security (CNS)*. IEEE. 2016, pp. 380–381.

[13] Longlong Li et al. “End-to-end network intrusion detection based on contrastive learning”. In: *Sensors* 24.7 (2024), p. 2122.

[14] Yunji Liang et al. “Behavioral Biometrics for Continuous Authentication in the Internet-of-Things Era: An Artificial Intelligence Perspective”. In: *IEEE Internet of Things Journal* 7.9 (2020), pp. 9128–9143. DOI: 10.1109/JIOT.2020.3004077.

[15] Ilya Loshchilov, Frank Hutter, et al. “Fixing weight decay regularization in adam”. In: *arXiv preprint arXiv:1711.05101* 5 (2017), p. 5.

[16] Jian Luo et al. “A multi-channel contrastive learning network based intrusion detection method”. In: *Electronics* 12.4 (2023), p. 949.

[17] Yuxiang Ma et al. “A balanced supervised contrastive learning-based method for encrypted network traffic classification”. In: *Computers & Security* 145 (2024), pp. 104023/1–12.

[18] Iaroslav Melekhov, Juho Kannala, and Esa Rahtu. “Siamese network features for image matching”. In:

2016 23rd international conference on pattern recognition (ICPR). IEEE. 2016, pp. 378–383.

[19] John V Monaco. "Device fingerprinting with peripheral timestamps". In: *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2022, pp. 1018–1033.

[20] David Monschein and Oliver P. Waldhorst. *mPSAuth: Privacy-Preserving and Scalable Authentication for Mobile Web Applications*. 2022. arXiv: 2210.04777. URL: https://arxiv.org/abs/2210.04777.

[21] Mark O'Neill et al. "TLS inspection: how often and who cares?" In: *IEEE Internet Computing* 21.3 (2017), pp. 22–29.

[22] Praveen Kumar Rayani and Suvamoy Changder. "Continuous user authentication on smartphone via behavioral biometrics: a survey". In: *Multimedia Tools and Applications* 82.2 (2023), pp. 1633–1667.

[23] Giuseppe Stragapede et al. "IEEE BigData 2023 Keystroke Verification Challenge (KVC)". In: *2023 IEEE International Conference on Big Data (BigData)*. IEEE. 2023, pp. 6092–6100.

[24] Giuseppe Stragapede et al. "Keystroke verification challenge (KVC): biometric and fairness benchmark evaluation". In: *IEEE access* 12 (2023), pp. 1102–1116.

[25] Giuseppe Stragapede et al. "KVC-onGoing: Keystroke Verification Challenge". In: *Pattern Recognition* 161 (2025), pp. 111287/1–14.

[26] Cheng Wang et al. "Behavioral authentication for security and safety". In: *Security and Safety* 3 (2024), pp. 2024003/1–36.

[27] Feng Wang and Huaping Liu. "Understanding the behaviour of contrastive loss". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 2495–2504.

[28] Yawei Yue et al. "Contrastive learning enhanced intrusion detection". In: *IEEE Transactions on Network and Service Management* 19.4 (2022), pp. 4232–4247.

[29] Bruno Bogaz Zarpelão et al. "A survey of intrusion detection in Internet of Things". In: *Journal of Network and Computer Applications* 84 (2017), pp. 25–37.