Two-factor, continuous authentication framework for multi-site large enterprises

Lehel Dénes-Fazakas
John von Neumann Faculty of Informatics
Obuda University
Budapest, Hungary
denes-fazakas.lehel@nik.uni-obuda.hu

Eszter Kail
John von Neumann Faculty of Informatics
Obuda University
Budapest, Hungary
kail.eszter@nik.uni-obuda.hu

Rita Fleiner
John von Neumann Faculty of Informatics
Obuda University
Budapest, Hungary
fleiner.rita@nik.uni-obuda.hu

***Abstract*— Security incidents originated from company employees or insiders give almost half of the attacks against organizations. Moreover, due to the pandemic situation, security mechanisms have a much higher role than ever. This paper proposes a two-factor authentication for multi-site large enterprises where continuous authentication, processing, and storage scalability and reliability are of primary importance. According to the requirements mentioned above, our solution is based on the Apache Spark framework extended with the Apache Cassandra, Kafka, and a MySQL database to provide data parallelism, fault tolerance, and distributed data storage.**

Keywords— authentication, distributed processing, Apache Spark, mouse dynamics

I. Introduction

According to Verizon's evaluation of data privacy incidents of health care during 2016 and 2017, more then half of the incidents originated from insiders **[1]**. Although during the last year outsiders' incidents of this sector superseded the ones originated from the insiders, insiders still cause nearly half of the incidents in other sectors, which is too high to ignore them. Also, the widespread usage of home office work and remote access to the company network raises security and authentication challenges. It can be concluded that safety requirements are now higher than ever. Thus in large organizations, it is essential to use strong authentication for the employees; at least two-factor or multi-factor authentication; even more, accounts should be authenticated and authorized continuously during their active sessions.

One factor authorization, (also called a knowledge-based factor) which is mostly based on some knowledge that a user owns, such as a password, or PIN is not considered secure enough anymore as an unauthorized user can also attempt to gain access by utilizing the dictionary attack [**2**], rainbow table [**2**], or social engineering techniques [**3**].

After realizing that one-factor authentication is not safe anymore due to the different security threats, a second factor was introduced that combined the so-called knowledge factor with the ownership factor (e.g., a smart card or a smartphone). The third "biometric" factor started to be involved in the authentication chain, which means having something unique that identifies people. Several different biometric-based authentication forms spread across the world like fingerprint-based, voice or facial recognition-based, or mouse dynamics-based authentication [**2**].

In this paper, the problem of two-factor and continuous authentication is investigated in a multi-site enterprise environment. When employees at first log into the system, a password-based authentication for the beginning of the session occurs, and then continuously analyzing the user's mouse movements through the active sessions, the authentication is verified from time to time. Our work's primary aim was to provide a secure, scalable, fast, and reliable framework for authentication in large enterprises. We investigated the technological solutions for building a distributed platform for parallel data processing, fault-tolerant big data storage, and scalability. In this work, we propose a plan for an authentication method that fulfills the requirements, we also present and evaluate our prototype solution working in a mono-site environment.

The structure of the paper is as follows. Section II. introduces related work in the topics of mouse dynamic based and continuous authentication and the used technological solutions. Section III. and IV. introduces our proposed and realized prototype system architecture. In section V. we briefly summarize the data collection, preprocessing, and user classification methods. Section VI. presents our feasibility results and, finally, in section VII. we conclude our work.

II. Related work

Mouse dynamics based user authentication is gaining more and more popularity. Mouse dynamics include monitoring the user behavior through how he/she interacts with the mouse as a means for authentication. Like other biometrics-based authentication [4], the way people move the mouse is unique; therefore, it is appropriate for authentication purposes. The key feature is that it enables dynamic and passive user monitoring while it does not need any new hardware to accomplish this as opposed to the fingerprint-based authentication where a fingerprint scanner is needed. Dynamic means that features extracted from the real movements can be adapted to the users' changing habits (when tired or changing with time), and passive means that users are not aware of it, they do not have to change their computer usage habits. Everyone uses a mouse or touchpad. It is one of the essential input devices when using PC-s or laptops. Moreover, passwords can be stolen through social engineering or lost, forgotten, but the way we move our mouse cannot be stolen so easily, it needs much practice.

Additionally, this authentication method enables continuous authentication during the mouse is moving. Most of the systems using authentication only at the beginning of a session, which raises security issues, since when an employee leaves his or her machine for a while, anyone can sit down and continue to work on his or her machine [5].

Several studies focus on mouse dynamics. Most of these solutions have two main phases. In the first phase, they gather data about the mouse's movements, and in the second phase, different features are extracted from the data and based on these features, the classification takes place. The result of the classification is generally a probability value of identifying a user. The data gathered is mostly composed of tuples describing a mouse's movement, including the timestamp, the velocity, the acceleration, the angle, the type of the movement.

In their research [6], Gamboa and Fred tried to find the answer to whether users can be recognized based on mouse interactions. They realized a memory game during which a javascript program gathers data about the position of the mouse. From these raw data, 63 features were extracted, and the classification took place. They did not use general mouse movements as they gathered data with the help of a unique memory game, but they managed to reach a fault rate of 2 % in the end.

Traore and Ahmed in [7] classified raw data into four classes (Mouse Movement, Drag and Drop, Point and Click, and Silence). From these 4 data types, they extracted statistical parameters. For each user, they used a binary neural network that was trained. The results were outstanding, but it cannot be used in real-time systems since they were generated after 17 minute-long data evaluation.

Zheng et al. [8] applied a new approach. They extracted the features from only one mouse movement. They called it angle based metric. The best results showed a fault rate of 1,3%, which is good, and their solution is especially significant since they needed very little data for making their decision. They used SVM (Support Vector Machine) as their decision-maker.

Ernsberger et al. in [9] realized a web browser extension to gather mouse movement data. After feature extraction, their system also visualized mouse movements, and in the end, their classification reached 78,1% accuracy.

Chong et al. in [10] used convolutional neural networks to extract features from mouse movement data. The input of the neural network was the two-dimension image of the mouse movement.

In our planned system, we intended to realize real-time mouse-dynamics based authentication; therefore, the planned system consists of the following open-source frameworks. For data collection and distribution, we applied the relevant state-of-the-art system Apache Kafka, for data processing, i.e., user authentication, the widely used open-source distributed general-purpose cluster-computing framework, Apache Spark, and for data storage, Apache Cassandra.

Apache Spark [11] is an open-source and widespread used cluster computing framework that was invented with the primary aim to accelerate and facilitate Big Data computations. It has emerged as the next generation big data processing engine, overtaking the Hadoop MapReduce role as it mainly stores data in memory, which enables much faster processing. It's most significant benefits stem from several easy-to-use Python, Java, Scala, and R's APIs. Moreover, it also supports various compute-intensive tasks beyond batch applications, including interactive queries, streaming, machine learning, and graph processing.

Apache Cassandra [12] is an open-source distributed NoSQL database management system. It was mainly invented to handle large amounts of data distributed across many commodity servers with the ultimate aim of providing a highly available service. Cassandra is a hybrid data management system between a column-oriented DBMS and a row-oriented store.

Apache Kafka [13] is the most popular framework used to ingest data streams into the processing platforms. It facilitates to get results continuously and in real-time from large volumes of data.

III. Architecture description

Our research's main objective was to design a platform-free 2-factor authentication system for multi location-based companies with more than thousands of employees, implement its prototype, and achieve feasibility tests on the implemented system. In this section, we describe the design and the system plan for a 2-factor authentication method, then we show the prototype developed according to our design.

The proposed 2-factor authentication system relies on a user providing a password as a first factor and the mouse movements (mouse dynamics) of the user as the second factor. The mouse dynamic authentication is achieved by machine learning techniques: it first learns the users' mouse movement features by building user-specific models, then it achieves continuous authentication on the users' mouse movement data. An important feature of our objective was to design a system for large company usage, where the number of users is not limited, the volume of the stored data at the server-side can reach the big data scale, and the data processing and storage are horizontally scalable. Therefore, we decided to consider a distributed system, where data replication is possible to provide high availability. Further requirements were modularity and stream-based data processing at the server-side.

The functional requirements of the server-side are the following:

- building user-specific machine learning models used for classification,
- classifying user feature vectors based on the previously built models,
- storing user feature vectors,
- storing username-password data,
- performing alerts due to detecting non-user specific mouse behavior,
- stream-based data processing,
- rebuilding the user-specific models using the users' most recent mouse dynamics data.

The functional requirements of the users' side are the following:

- extracting feature vectors from mouse movements,
- sending these vectors to the server.

The dataflow in the designed 2-factor authentication system is depicted in Fig 1.

We propose to use the Apache Spark framework [11] for data processing and for iterative machine learning tasks, Apache Cassandra [12] for storing the users' mouse movement feature vectors, and Apache Kafka [13] for the stream-processing component.

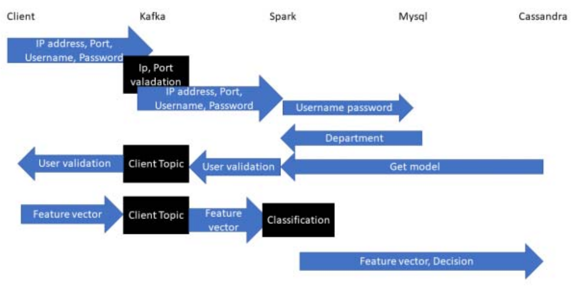

Fig. 1. Dataflow of the 2-factor authentication system

The main components of the designed system are the following as shown in Fig. 1.: Apache Kafka, Apache Spark, MySQL database, and Apache Cassandra NoSQL database. (1) The client program at the user's side records the mouse movements' specific details, creates feature vectors from these mouse movement points, and sends them as a producer to Kafka. (2) In the Kafka component, the user acts as a producer and Spark as the consumer. More than one broker is proposed to achieve a higher replication factor and data processing distribution. There will be one Kafka topic for the IP address, PORT number, username, and password coming from any user. Furthermore, a unique topic will be created for each user to store the user's mouse movement feature vectors. (3) The Spark component's task is to process the feature vector data coming from the users, perform the authentication of the user based on the incoming feature vectors, and send the incoming streaming data to Cassandra. (4) MySQL database stores the first- factor authentication data (username, password, department) in a centralized way. (5) The user-specific ML model (trained with the user's previously sent mouse dynamic data), the feature vectors, and the result of the user's classifications are stored in a distributed manner in Cassandra database nodes. Cassandra can handle the replication of the data to provide redundancy for the sake of availability and reliability.

The dataflow in the system is the following (see Fig. 1.). The user sends the username and password as the key, and IP address and PORT number as the value to Kafka. Spark, acting as a consumer, reads this data and sends this username and password values further to the MySQL database. MySQL stores the initially approved username and password data. After successfully authenticating the user by username-password, the user is continuously sending the mouse dynamic data (feature vectors) through Kafka to the Spark component. The user-specific model (based on the mouse movement data previously collected from the client) is stored in the Cassandra database. After authenticating the client by username-password, Spark reads in the user-specific model from Cassandra and continuously checks the user's identity by evaluating the model with the presently sent mouse feature vector data.

IV. Prototype

We developed the prototype (see in Fig. 2.) of the above-suggested design and accomplished feasibility tests based on publicly available mouse dynamics data set. In the research, we followed the implementation guidelines of the system-component software and tried to implement the main features in Python. However, some actions are not yet supported in Spark with Python, thus upon this process, we faced some bottlenecks which have to be considered and changed in the future development of the entire system.

The client program at the user side is written in Java. The client sends the IP, PORT, username, and password values to Kafka, where the IP is validated. Spark reads these values from Kafka, authenticates the user by checking the username-password validity in the MySQL database, gets back the user's department number from MySQL, and creates a dedicated Spark job for this user. The Spark job also reads in the user-specific model from the Cassandra database. Through the connection between the Spark job and the user, Spark reads the feature vectors as streaming data from the users directly. In the Kafka environment, one broker with one topic is used. The topic contains the username and password as the key, and IP address and PORT number as the value. In Cassandra, data is distributed among 4 nodes, replication is not used at this point. The dataflow mechanism is depicted in Fig 3.

In our feasibility test, every 5 seconds 10 feature vectors were produced by the users. Spark achieves retraining of the users' models with the most recently sent feature vectors. The retraining process happens once per month or after sending one thousand feature vectors.

V. Data set

A. *Data collection*

The data collection for our dataset was carried out with a platform free application written in Java. The application is operating in server mode and is activated upon operating system initialization. The data collection is transparent and unnoticed by the user. During initialization, a .csv file is created where each row represents one mouse movement as a form of a 5-tuple.

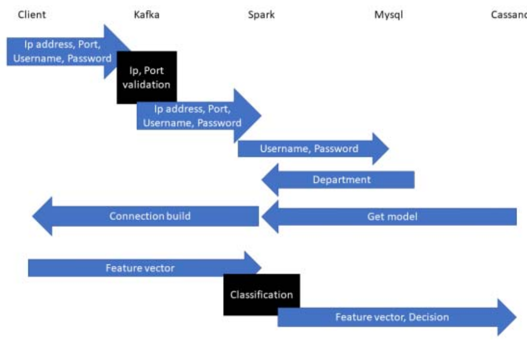

Fig. 2. The architecture of the prototype of the 2-factor authentication system

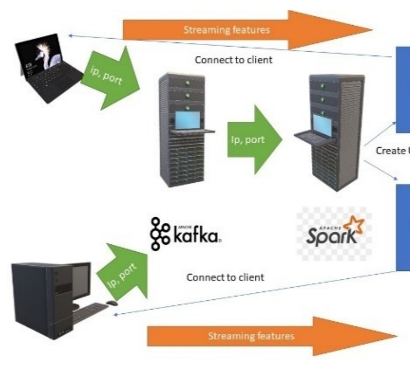

Fig. 3. Dataflow in our prototype authentication system

These tuples consist of the client timestamp, button type (Left, Right, NoButton), state (Move, Pressed, Released, Drag), and x,y coordinates. For the collection of the data, 21 subjects (15 men and 6 women) were involved in an uncontrolled environment, who carried out their daily activities on their devices. The data set is publicly available [14] and can be downloaded.

B. *Preprocessing*

After collecting the data, a 2-step preprocessing was used.The raw data at first should be cleaned in order to eliminate invalid or wrong data. Such uninterpretable data consist of rows in the .csv file where the datestamps were the same, or where the different tuple values were not in the valid range. The cleaned mouse movements tuples were segmented in the second phase into meaningful mouse actions which could be clearly separated. In our work, we extracted 23 different features from each action. These features include different distance -, speed-, acceleration -, angle parameters, button type, action type, etc., which is a subset of the specified features mentioned in [6], [15], [16], [17].

C. *User classification*

A binary classifier was used for each user based on valid user (investigated subjects') generated mouse movements and invalid user movements from the outside world. 70 % of the collected data were used as the training set to train the model, and 30% were test data.

During our preliminary tests, we compared the performance of 4 different classifiers (Logistic regression, Random Forest, Neural networks, and K-means) on a well-balanced data set, which consisted of 1000 positive and 1000 negative samples. For evaluation, we used ROC curves, accuracy, and AUC value for the individual subjects. The best performance could be observed by the Random Forest method. Based on these results, our prototype solution also applies the Random Forest algorithm for user classification. The main parameter of this algorithm is the number of trees. To determine this number, we increased the number (n) from 10 to 100 while evaluating user classification accuracy. The best accuracy was reached with n=80. Above this value, the improvement was not significant, so we decided to use 80 trees in our algorithm.

We also carried out experiments for determining the appropriate number (N) of consecutive mouse movements that could serve as an input for the classifier to reach the highest accuracy. According to our observations, increasing the number causes better performance, and N=10 gave satisfactory results. We used the average of 10 prediction results and applied a sliding window technique where the window was moved forward by one value in each step. The alarm triggering action required that the above-mentioned average falls below a predefined *p* probability.

VI. Experiments and Results

In our experiments, we conducted feasibility tests to prove the efficiency of the proposed architecture. At first, we tested the time for initializing the system and compared the loading time for the user's model from MySQL, Cassandra, and also from a file. From the output, it can be revealed that the worst case is when all the models are stored in MySQL since its performance is varying and needs much more time. Although Cassandra's reading takes a bit more time than reading from a file, the reliability is the highest. Table 1. shows the measured and computed metrics for the different read-in types in a scenario where 9 consecutively logged in users were in the system.

In the second experiment, we tested the classification time and the effect of the increasing number of users in the system. In this test, 14 users logged into the system sequentially with minimal delays. Table 2. shows that the number of the users in the system does not affect the classification performance, and in general, it can be concluded that the system's response time is less than the streaming interval, which is 5 seconds in our case.

VII. Conclusion

In this paper, we proposed a 2-factor and continuous authentication framework for multi-site, large enterprises. We implemented a prototype system with just one site. The first stage of the authentication is based on username password validation, and after logging in the system, user activity is continuously monitored and revalidated with the help of a mouse-dynamics based Machine learning subsystem. The architecture was designed with the ultimate goal of providing a fault-tolerant and scalable, real-time processing authentication framework. Therefore, we applied the Apache Spark computing framework, Apache Kafka, for supporting real-time processing tasks and Apache Cassandra distributed database management system. We have conducted feasibility tests for proof of concept validation. The results showed the system's benefits, like reading input data and stream data quickly and efficiently, and confirmed that this authenticating system could work smoothly with real-time collected and preprocessed data.

In the future work we plan to exploit the distributed processing features of Spark by using more Spark nodes, to increase the reliability by using replication in Cassandra, and to move this architecture to the cloud.

TABLE I. Loading times of the model (msec)

	AVG	STD	MIN	MAX
Cassandra	0.04803	0.02356	0.01855	0.22041
MySQL	1.81578	7.60228	0.4797	72.8310
file	0.012867	0.00602	0.0089	0.05485

TABLE II. CLASSIFICATION AND STORING TIME

	mean	std	min	max
1	1.8561	1.3859	0.2723	9.248
2	1.9914	1.4085	0.3296	9.1325
3	2.0541	1.231	0.4997	7.623
4	2.233	1.2871	0.6513	7.35
5	2.2008	1.2402	0.8697	8.4072
6	2.2276	1.1237	0.7561	5.9446
7	2.3638	1.5344	0.3092	9.4325
8	2.4991	1.2421	0.8553	6.4561
9	2.6413	1.628	0.9106	8.6918
10	2.4766	1.3052	0.8792	7.6671
11	2.5098	1.5217	0.7859	8.6201
12	2.6275	1.2942	0.8001	7.6972
13	2.4648	1.5239	0.8369	7.5302
14	2.3677	1.4225	0.3312	7.8661

REFERENCES

[1] "New report puts healthcare cybersecurity back under the microscope", 2020. [Online]. Available: https://www.verizon.com/about/news/new-report-puts-healthcare-cybersecurity-back-under-microscope. [Accessed: 09- Oct- 2020].

[2] A. Ometov, S. Bezzateev, N. Mäkitalo, S. Andreev, T. Mikkonen and Y. Koucheryavy, "Multi-Factor Authentication: A Survey", Cryptography, vol. 2, no. 1, p. 1, 2018. Available: 10.3390/cryptography2010001.

[3] R. Heartfield and G. Loukas, "A Taxonomy of Attacks and a Survey of Defence Mechanisms for Semantic Social Engineering Attacks", ACM Computing Surveys, vol. 48, no. 3, pp. 1-39, 2016. Available: 10.1145/2835375.

[4] Zhang, Xuantong, "Human user authentication based on mouse dynamics: a feasibility study" (2015). Graduate Theses and Dissertations. 14880.

[5] Salman, O. and Hameed, S., 2020. User Authentication Via Mouse Dynamics. [online] Iasj.net. Available at: https://iasj.net/iasj/article/146207.

[6] Hugo Gamboa and Ana Fred. A behavioral biometric system based on human-computer interaction. In Biometric Technology for Human Identification, volume 5404, pages 381–393. International Society for Optics and Photonics, 2004.

[7] Ahmed Awad E Ahmed and Issa Traore. A new biometric technology based on mouse dynamics. IEEE Transactions on dependable and secure computing, 4(3):165–179, 2007.

[8] Nan Zheng, Aaron Paloski, and HainingWang. An efficient user verification system using angle based mouse movement biometrics. ACM Transactions on Information and System Security (TISSEC), 18(3):11, 2016.

[9] Ernsberger Dominik, VSH Ikuesan Richard Adeyemi, and Alf Zugenmaier. A web-based mouse dynamics visualization tool for user attribution in digital forensic readiness. In 9th EAI International Conference on Digital Forensics & Cyber Crime, 2017.

[10] P. Chong, Y. X. M. Tan, J. Guarnizo, Y. Elovici, and A. Binder. Mouse authentication without the temporal aspect what does a 2d-cnn learn? In 2018 IEEE Security and Privacy Workshops (SPW), pages 15–21, May 2018.

[11] Spark.apache.org. 2020. Overview - Spark 2.4.5 Documentation. [online] Available at: https://spark.apache.org/docs/2.4.5/T.

[12] Cassandra.apache.org. 2020. Documentation. [online] Available at: https://cassandra.apache.org/doc/latest/.

[13] Apache Kafka. 2020. Apache Kafka. [online] Available at: https://kafka.apache.org/documentation [Accessed 8 October 2020].

[14] THE DFL MOUSE DYNAMICS DATA SET Your Bibliography: Ms.sapientia.ro. 2020. The DFL Mouse Dynamics Data Set. [online] Available at: https://ms.sapientia.ro/~manyi/DFL.html .

[15] ANTAL, Margit; DENES-FAZEKAS, Lehel. User Verification Based on Mouse Dynamics: a Comparison of Public Data Sets. In: *2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. IEEE, 2019. p. 143-148

[16] Zaher Hinbarji, Rami Albatal, and Cathal Gurrin. Dynamic user authentication based on mouse movements curves. In Xiangjian He, Suhuai Luo, Dacheng Tao, Changsheng Xu, Jie Yang, and Muhammad Abul Hasan, editors, MultiMedia Modeling, pages 111–122, Cham, 2015. Springer International Publishing.

[17] Réman Krisztina, Dénes-Fazakas Lehel, TDK-dolgozat, Sapientia EMTE, XVII. TDK, 2018.