NLP-based User Authentication through Mouse Dynamics

Hoseong Asher Lee[1], Nikhil Prathapani[1], Rajesh Paturi[1], Sarp Parmaksiz[1] and Fabio Di Troia[2][a]
[1]*Department of Computer Engineering, San Jose State University, San Jose, CA, U.S.A.*
[2]*Department of Computer Science, San Jose State University, San Jose, CA, U.S.A.*

Keywords: Insider Threat Attack, Intrusion Detection, CNN, LSTM, biLSTM, NLP.

Abstract: Insider threat attacks are increasing in most organizations yearly. It is also tough to prevent this type of attack because the threat is within the boundary, making them more dangerous than external threat actors. There can be a situation where a strong authentication layer is implemented for the external users, but due to cost or maintenance effort reasons, the authentication layer for insiders might not have proper security controls. One of the types of insider threat attacks is to exploit established sessions by legitimate users. There are certain applications and operating systems that provide an in-built security mechanism to detect idle sessions and automatically expire the sessions if no action is performed by the user. However, this type of protection is still vulnerable since it cannot really detect if the user who is taking action is the legitimate user or not. In this paper, we propose to use an advanced machine learning model based on Natural Language Processing (NLP) algorithms to authenticate users based on their mouse dynamics in web browser contexts. The model can provide a protective layer that continuously monitors against insider threat attacks. By this method, we can prevent malicious users from accessing unauthorized assets and provide enhanced security to legitimate users.

1 INTRODUCTION

The internet technology has been increasingly adopted to our daily lives. From smartphones to the technologies allowing us to 'work-from-home', we are in an era where digital information is widely exposed. All this led to the field of cybersecurity becoming more expansive. Today, cybersecurity is needed and widely adopted than it was in the recent past. In computers, we have firewalls, antivirus software, intrusion detection and prevention technologies that protect users and companies from outside threats. There are also operating system level protections, such as authentication that might prevent an attacker from taking over a machine. Mac OS uses Keychain (Apple Inc., 2021) for authentication, while Linux systems use Pluggable Authentication Module (PAM) (RedHat, 2020) to store authentication details. However, there is not much security after a user is authenticated, which is where an insider threat damages the most. The lack of internal security provides a challenge in congregated settings, such as schools, colleges, corporate organizations, and government offices. In these types of situations, users need a solution that helps them to protect their browsing data, which might contain sensitive financial information, intellectual property, and personally identifiable information. Machine learning can be potentially a promising solution for such problems. In the internet browsers, web applications or browser extensions can track the end-user's mouse dynamics, and that data can be used to train a machine learning model to authenticate the user. As the model gets trained and more mature, both the security and the user experience are greatly improved. The end-users do not need to periodically authenticate themselves, rather, they will be prompted for authentication only when the model suspects the end-user. Also, this can prevent malicious users from intercepting the victim's browser and stealing data from it.

The remainder of this paper is organized as follows. In Section 2, we describe the state-of-the-art in the intrusion detection field by briefly introducing related work in this area. In Section 3, we discuss background topics such as the machine learning models adopted in this research. Section 4 covers the applied methodology analyzing in detail every specific approach followed for every machine learning model. In Section 5, we present and analyze the results of

[a] https://orcid.org/0000-0003-2355-7146

Lee, H., Prathapani, N., Paturi, R., Parmaksiz, S. and Di Troia, F.
NLP-based User Authentication through Mouse Dynamics.
DOI: 10.5220/0011005900003120
In *Proceedings of the 8th International Conference on Information Systems Security and Privacy (ICISSP 2022)*, pages 696-702
ISBN: 978-989-758-553-1; ISSN: 2184-4356

our experiments. Section 6 describes the software implementation of our proposed work by introducing a web-browser extension. Finally, Section 7 presents the conclusions of our paper and includes a discussion of possible directions for future work.

2 RELATED WORK

Several works have been piblished on the long lasting problem of intrusion detection. For example, the work in (Saber, 2019) describes a machine learning-based approach to detecting vehicle theft by analyzing the anomalies in the driving behavior of the user. Another example is the work in (Manikoth et al., 2018), where the authors implemented a detection mechanism based on several classifiers with the goal to find the best subset of features to identify unauthorized use of a mobile device. In (Huang et al., 2021), the authors propose a new dataset to authenticate users to their own mobile devise. This last work relies on gesture-based authentication, where the sensors informantion are used to train machine learning models. As seen in this examples, in order to achieve machine learning-based user authentication, there can be different approaches to accomplish it. We describe now different of such approaches proposed in academia over the past few years. In particular, we concentrate on browser intrusion and insider threat attacks. Three works step forward due to their promising results and innovative solutions in the field of intrusion detection. One approach was to intervene with users while they are using the computer periodically (Chen et al., 2014). The approach was proven to be successful in terms of achieving high accuracy to predict a legitimate user. For five seconds of verification, it achieved 2.86% False Rejection Rate (FRR) and 4.00% False Acceptance Rate (FAR). Where, FAR is the percentage of identifications in which unauthorised individuals are incorrectly accepted (also called fraud rate), while FRR is the percentage of identifications in which authorised individuals are incorrectly rejected (also called insult rate). Another method was to extract features from mouse operations (Jorgensen and Yu, 2011). This approach was also fairly successful, resulting in roughly 2% of both FRR and FAR. However, the average time to authenticate users was relatively long, for example several minutes to sometimes even more than 10 minutes. Although it can predict a malicious user, within that time frame, the malicious user can still performs a considerable amount of damage to the victim's data. Hence, the fact that it takes too long to authenticate users makes this approach not feasible to be applied in the real world. The work proposed in this paper takes inspiration from a research which proposed to use CNN algorithm to authenticate users using mouse dynamics (Hu et al., 2019). In this approach, the authors converted mouse operations into JPEG images following certain rules. They used an open-sourced dataset for mouse dynamics. And with that, they were able to achieve 2.96% FAR and 2.27% FRR within only seven seconds. The great advantage of this approach is that it does not extract features from a model, and does not miss any information from the user actions. Furthermore, it does not require any other algorithms to extract features from the dataset. For this reason, we decided to improve further in the direction undertaken by this interesting work. In particular, we decided to apply Natural Language Processing (NLP) algorithms to the extracted user data.

3 BACKGROUND

In this Section, we describe different types of machine learning algorithms that we applied for mouse dynamics user authentication in our experiments

3.1 CNN

Convolutional Neural Network (O'Shea and Nash, 2015) (CNN) is a type of neural network algorithm that is popular for analyzing images and detecting patterns. One of the characteristics of the network is that it has multiple convolutional layers, and those layers are responsible for finding patterns in input images. One can think of an input image as a representation of a mathematical matrix. CNN internally applies filters to groups of cells in the matrix, and this results in the pattern recognition. There are many types of filter for recognizing different patters, for instance simple shapes such as squares or edges. By utilizing those filters, more sophisticated shapes can be detected, such as a dog, a cat, or a human's face. A representation of the patterns recognized by a CNN is shown in Figure 1 for different classified objects.

3.2 LSTM

LSTM stands for long short term memory and it is a type of recurrent neural network (RNN) algorithm. Traditionally, RNN had an issue of short term memory where important data could not be propagated to the final layers during prediction. LTSM aims to reduce this problem. RNNs usually have multiple short term memory cells. In the case of LSTM, we have a memory cell which includes scope for long term

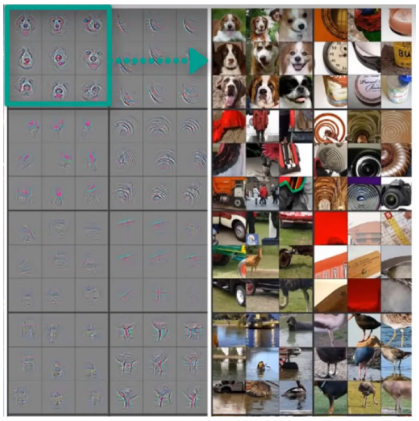

Figure 1: The left side panel contains the output result from a CNN model, and the output results contain the patterns identified from the input pictures on the right side panel (Dshahid380, 2019).

memory and short term memory. For example, if we have an input word in short term memory operation, it is converted into a list of numbers or a vector $x(t)$, while previous hidden state is denoted by $h(t-1)$, which is also a vector. Finally, a sigma operation is computed as weighted multiplication, and the activation function hyperbolic tangent (tanh) is applied. During LSTM training, a large number of statements are inputted to the algorithm which will build the understanding in the recurrent neural network. Throughout this process, the model discards unnecessary information and stores only what is relevant. A full LSTM operation comprises of a forget gate, input gate, and output gate, as denoted in Figure 2. As the name suggests, forget gate is used to discard information that is not needed for prediction in future. In the case of a forget gate, the previous hidden state $h(t-1)$ and the current input $x(t)$ are activated through a sigmoid function that restricts the value between 0 and 1. Finally, the output of the sigmoid function is multiplied with the previous memory state $c(t-1)$.

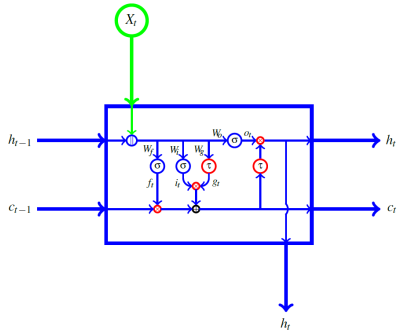

Figure 2: LSTM gates and activation functions (Dang et al., 2021).

In the case of the input gate (that adds new memory-meaningful information), we have sigmoid and tanh functions both for "h(t-1)" and "x(t)", and each vector is bound with a weight when is given to the function. Both of these outputs are multiplied together, and long term memory data is then added. Finally, we have the output gate, where a weighted sum of $h(t-1)$ and $x(t)$ is computed before being activated by the sigmoid function. This computed output is then processed to generate a new hidden state $h(t)$.

3.3 biLSTM

Bidirectional long-short term memory (biLSTM) is the successor of the unidirectional LSTM model. In regular LSTM, the only information that is present is the input from the past. Unlike unidirectional LSTM, biLSTM can take inputs from both the past and the future of the input sequence relatively to a given word. This architecture style makes biLSTM more versatile than the LSTM model. According to the study in (Abduljabbar et al., 2021), when applied to the same dataset, biLSTM outperformed unidirectional LSTM by a large margin. Another study (Siami-Namini et al., 2019) also proved that biLSTM is around 38% more accurate than regular LSTM based on the dataset under analysis.

4 METHODOLOGY

We implemented several experiments using CNN, LSTM, and biLSTM. Here we give a description of the methodology followed to accomplish such experiments.

4.1 CNN

Overall, this approach comprises of three different steps:

1. Conversion of CSV data to JPEG Images
2. Data Augmentation
3. Model Training

For the most part, we adopted the methods proposed in (Balabit, 2021), where the mouse dynamics dataset was taken from an open source data repository. We adopted the same dataset. Since the repository provides the data in the CSV format, the data needs to be converted before being fed to the CNN algorithms. Hence, we converted the CSV data into JPEG images. The mouse operation is mapped to certain different shapes of drawing in the JPEG image. Table 1

shows the different input actions and the corresponding graphical representation. Each JPEG image contains a certain number of mouse operations, and the number can vary by choice. In our study, we experimented with 100, 500, and 1000 mouse actions.

The CNN model contains seven layers. The first four layers are convolutional layers, and the remaining three are fully-connected layers. The ReLU activation function was applied to all layers in the model except for the last one. The input shape to the first convolutional layer was set to $[100, 100, 3]$. The first two values indicate the width and height of input JPEG, and the last value indicates the color channel. Hence, the model reads images with red, blue, and green colors with 100 pixel width and 100 pixel height. The kernel size was set to 3 for all convolutional layers. The filter was set to 32 for the first convolutional layer, 32 for the second convolutional layer, 64 for the third convolutional layer, and 128 for the last convolutional layer. A max pooling layer was also added after each convolutional layer.

4.2 JPEG Image Data

The open source dataset consists of 10 different users' mouse dataset. An image rotation method to augment the data was also implemented in our experiments. Out of the converted JPEG images, we picked random images and rotated them by 90, 180, or 270 degrees. The selection of the angle was also random, and the augmentation was accomplished until it reaches the maximum number of images.

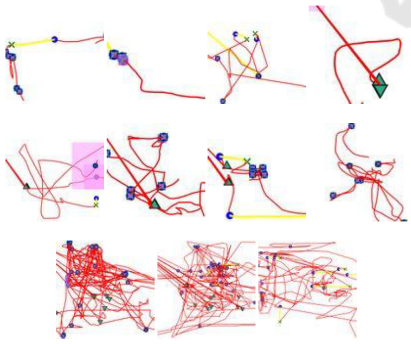

Figure 3: Converted JPEG images from CSV-formatted mouse operation data (Balabit, 2021). Each row of images is created with different mouse actions (m) values. From the top row, m is set to 100, 500, and 1000, respectively.

4.3 LSTM

For LSTM , we tested three different approaches:

1. Model training with basic GRU (Gated Recurrent Unit)
2. Unidirectional LSTM
3. Bidirectional LSTM

For all three approaches, we first began by reading the input dataset as numpy files. Numpy files in python provide support for large arrays and matrices. We also added an additional script to read the contents of numpy data files. After we loaded the inputs as numpy files, the next step was to choose either GRU, unidirectional LSTM or biLSTM for model training. After we compiled a model, the next step was to fit the model. Here, we chose the X value and Y value from input numpy arrays. Also, we chose the number of epochs, which basically denotes the number of passes of the entire training dataset that the ML model has completed. We also chose the validation split which is the parameter specifying how much of training data is used for validation (a value between 0 and 1). Validation data is not used for the training, but to evaluate the loss and the accuracy. In our work, we set "validation_split=0.1", that is, 10% of the training data was used for validation. Next step after fitting the model was to train the model and obtain the results of accuracy and loss over various epochs. We also plot the confusion matrix for the same. Confusion matrix is used for performance measurement in machine learning classification where the output is a table with four different combinations of predicted and true values. It is utilized for recall, accuracy, and precision of a model. The four possible outcomes are True positive, True negative, False positive, and False negative. True positive means that you predicted positive and the outcome was correct. True negative means that the model predicted negative and the outcome was not correct. False positive means that the model predicted positive and the outcome was not correct (type 1 error). False negative means that the model predicted negative and the outcome was not correct (type 2 error).

5 RESULTS

We trained the models with different machine learning algorithms. Here, we describe the outcomes of our experiments in detail.

For the fully-connected layers, the first layer contains 1024 neurons passing Dropout with probability of 50%, and the second layer contains 512 neurons passing Dropout with probability of 50%. The Adam optimizer function was used with the parameters learning_rate=0.01, beta1=0.9, beta2=0.999, and epsilon=1e-08. To train the model, the legal user data and illegal user data were mixed randomly.

Our first experiment relied on purely CNN as a confirmation of the work in (Hu et al., 2019), which

Table 1: Mapping between mouse operations and corresponding drawings.

Operation from CSV data	Mapped Drawing in JPEG image
Move	Red Line
Pressed	Blue Circle
Released	Green ”X” Shape
Drag	Yellow Line
Scroll Up	Cyan Upward Triangle
Scroll Down	Cyan Downward Triangle
Stay	Red Translucent Square (Size of Square related to the time of Stay operation)

obtain a maximum accuracy of around 81%. The meaning of accuracy in this context is how often the anti threat system was able to detect a malicious intruder. Table 2 contains the results of this experiment with different values of mouse operations (m).

Then, we tested the GRU model, which was able to achieve a maximum accuracy of around 75%. Table 3 contains the model parameters and layers information. Please note that, for this and the subsequent experiments, there was no need to convert the data to images. This reduced considerably the overhead in analyzing the user's behavior.

Unidirectional LSTM, instead, obtained a maximum average of around 78%. Table 4 describes the parameters and layers used. Both values from LSTM and GRU were below the results obtained in (Hu et al., 2019), even though by a realtively small margin. However, implementing the biLSTM layer, the best accuracy achieved was around 95%, a considerably superior result when compared to the previous work. Table 5 describes the parameters and layers used. By these results, we see that biLSTM is a valuable and promising algorithm to implement when insider threat attacks are a possibility. A comparison of all the four approaches is given in Figure 4.

In Section 6, we propose a possible real-life implementation of this approach as part of a browser extension.

6 IMPLEMENTATION

As mentioned earlier, the mouse dynamic authentication can be used in a browser extension. Through the content script of a browser extension, the tool can access the mouse operations data for end-users and provision a ML model for such user. To achieve this, the user's mouse operation are collected. Those collected data is then used for both training the ML model and for authenticating the end-user in real-time. The project architecture follows a typical browser extension architecture where the browser extensions run asynchronously. One possible way to implement such architecture is to not develop a backend component. In this way, the machine learning model can be deployed in a backend server and the browser extension can leverage the model in the backend. However, with this design choice, it would incur in constant backend server maintenance. Also, if hackers run a DDoS (Distributed Denial of Service) attack against the central server, the authentication mechanism on the browser extension can be impacted. Hence, for scalability and security perspective, we decided to run the model in the browser without contacts with the server. This is possible with the employment of TensorFlowJS (Google, 2021). TensorFlowJS allows developers to run machine learning models and algorithms in the browser context using the JavaScript programming language. The browser extension runs asynchronously in the browser, and the browser collects and analyzes the mouse dynamics from the end-users by using Browser DOM API and TensorFlow models. If the extension concludes that the end-user is not the actual user but a malicious user, it locks the browser and the user is not able anymore to use it until they solve the authentication challenges. For instance, an employee can have specific access to certain internal resources, such as app development timelines or codebases. Those resources are considered confidential in most cases because they can affect the business operation if they are leaked. In the insider threat attack scenario, an employee would have been already authenticated for accessing a resource and the session is active in the browser. The employee could leave the computer unattended at risk of a bad actor taking actions on such device. Specifically, the malicious insider could access the resources and steal them. However, with the use of the browser lock ex-

Table 2: CNN model prediction by the number of mouse operations (m).

m	Accuracy	Loss	# of Legal User data	# of Illegal User data
100	0.811	3.04	4219	18054
500	0.810	3.05	850	3617
1000	0.809	3.06	426	1807

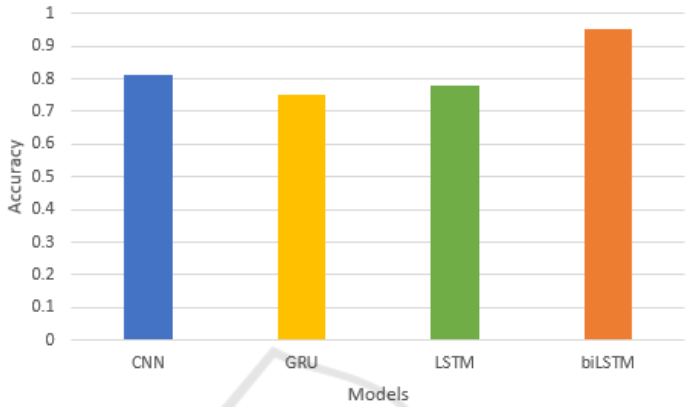

Figure 4: Maximum accuracy of the four tested model over mouse actions values of 10, 100, and 1000, obtained after 10 training epochs.

Table 3: Model parameters and layers for the GRU model.

Layer (type)	Output Shape	Parameters
input_1 (InputLayer)	(None, 100, 9)	0
gru (GRU)	(None, 6)	306
dense (Dense)	(None, 2)	14

Table 4: Model parameters and layers for the LSTM model.

Layer (type)	Output Shape	Parameters
input_1 (InputLayer)	(None, 100, 9)	0
lstm (LSTM)	(None, 6)	384
dense (Dense)	(None, 2)	14

Table 5: Model parameters and layers for the biLSTM model.

Layer (type)	Output Shape	Parameters
input_1 (InputLayer)	(None, 100, 9)	0
biderectional	(None, 12)	768
dense (Dense)	(None, 2)	26

tension, the browser can detect the hacking attempt by analyzing the mouse dynamics and prevent insider threat attacks. When the browser is locked due to suspicious attempts, the end-user is given an authentication challenge, and only if the user solves such challenge, the browser will be unlocked. The challenge can vary based on the implementation, but it can even be a traditional password authentication.

7 CONCLUSIONS

As a part of this research, we worked with mouse dynamics data to tell malicious users behavior apart from legitimate users activity. At first, we used a CNN model where we converted a sequence of mouse movements belonging to given users to JPEG images, and use the images as input to train a CNN model for user authentication. With the CNN model, we were able to achieve user authentication with about 81% of accuracy. Furthermore, we tested GRU, unidirectional LSTM, and biLSTM. With a single GRU layer after 10 epochs, the best accuracy was around 75%. With a single LSTM layer after 10 epochs, the best accuracy achieved was around 78%. With the biLSTM layer and after 10 epochs, the best accuracy achieved was around 95%. At the outset, the biLSTM model result was very promising, and it might be ideal to use this type of model for real-world applications like browser lock extensions where we identify malicious insider attacks.

As future work, we want to use a unified dataset where we can compare biLSTM to hybrid CNN-LSTM and CNN-biLSTM based approaches. Moreover, the dataset used could be enhanced with additional users and data augmentation techniques to test even further the efficacy of the proposed biLSTM method.

REFERENCES

Abduljabbar, R. L., Dia, H., and Tsai, P.-W. (2021). Unidirectional and bidirectional lstm models for short-term traffic prediction. *Journal of Advanced Transportation*, 2021.

Apple Inc. (2021). What is Keychain Access on Mac? https://support.apple.com/guide/keychain-access/what-is-keychain-access-kyca1083/mac. Online; accessed November 2021.

Balabit (2021). Mouse-Dynamics-Challenge. https://github.com/balabit/Mouse-Dynamics-Challenge. Online; accessed November 2021.

Chen, X.-j., Xu, F., Xu, R., Yiu, S.-M., and Shi, J.-q. (2014). A practical real-time authentication system with identity tracking based on mouse dynamics. In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 121–122. IEEE.

Dang, D., Di Troia, F., and Stamp, M. (2021). Malware classification using long short-term memory models. *arXiv preprint arXiv:2103.02746*.

Dshahid380 (2019). Convolutional Neural Network. https://towardsdatascience.com/covolutional-neural-network-cb0883dd6529. Online; accessed November 2021.

Google (2021). TensorFlow.JS. https://www.tensorflow.org/js. Online; accessed November 2021.

Hu, T., Niu, W., Zhang, X., Liu, X., Lu, J., and Liu, Y. (2019). An insider threat detection approach based on mouse dynamics and deep learning. *Security and Communication Networks*, 2019.

Huang, E., Di Troia, F., Stamp, M., and Sundaravaradhan, P. (2021). A new dataset for smartphone gesture-based authentication. In *ICISSP*, pages 771–780.

Jorgensen, Z. and Yu, T. (2011). On mouse dynamics as a behavioral biometric for authentication. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 476–482.

Manikoth, S. N. K., Di Troia, F., and Stamp, M. (2018). Masquerade detection on mobile devices. In *Guide to Vulnerability Analysis for Computer Networks and Systems*, pages 301–315. Springer.

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

RedHat (2020). An introduction to Pluggable Authentication Modules (PAM) in Linux. https://www.redhat.com/sysadmin/pluggable-authentication-modules-pam. Online; accessed November 2021.

Saber, A. (2019). Masquerade detection in automotive security.

Siami-Namini, S., Tavakoli, N., and Namin, A. S. (2019). The performance of lstm and bilstm in forecasting time series. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 3285–3292. IEEE.