Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA TELEKOMUNIKACJI

Praca dyplomowa inżynierska

Securing Web applications using methods of behavioral biometrics

Zabezpieczanie aplikacji WWW za pomocą metod biometrii behawioralnej

Autorzy:	*Kamil Kaliś, Piotr Kuglin*
Kierunek studiów:	*Elektronika i Telekomunikacja*
Typ studiów:	*Stacjonarne*
Opiekun pracy:	*dr hab. inż. Piotr Chołda*

Kraków, 2021

We hereby express our thanks to our supervisor – Piotr Chołda, PhD, who was guiding the process of creation of this thesis. The support in granting the needed resources for the study and feedback provided by Mr. Chołda are invaluable.

Special thanks to Krzysztof Rusek, PhD – for consulting the approach for proper machine learning model execution.

This research was supported in part by PLGrid Infrastructure.

Contents

1. Introduction

1.1. Abstract

Cyber security is a hot topic, especially nowadays, when there are many threats and vulnerabilities in hardware as well as in software and the facilitated remote access to computers. In the era of coronavirus, many people work from home. This increases the risk of attacks because they are outside the internal company network and many of them use their own devices, thus the remote session could be easily hijacked. People store sensitive data on their phones and other devices exposing them to a potential risk of stealing, extortion, or blackmail. Therefore, security is necessary and nowadays it is developed and discussed dynamically in various applications and fields. One of them is invisible user authentication that identifies the user based on his behavior, such as keystroke dynamic or mouse dynamics. The mentioned concept is relatively new because it requires a lot of computational resources and therefore could not be developed in the past. The evolution of technology and machine learning approaches, such as deep learning, allows us to imagine behavioral biometrics as a possible future solution for securing applications and hardware. The idea is fresh and therefore gives the potential to conduct the research. However, it requires additional effort to collect the proper dataset for the machine learning model. The problem raised in this thesis concerns distinguishing the malicious traffic generated by bots on the websites. There exist some public datasets mentioned in related works such as the Balabit dataset[4*], but they are not suitable for the problem of distinguishing a bot from a human. Therefore, the main ambition of this work is the creation of a system to gather such data from an example website, alongside the creation of a bot that impersonates the human-like user. The authors proposed also a machine learning model basing on the collected data.

This work consists of several chapters. In the beginning, the authors describe the main concept (Chapter 1) of the conducted research alongside the related works that are an inspiration for the presented thesis. To understand the concepts raised in this work, the brief theory for the main topics is introduced (Chapter 2). The further chapter describes an implementation (Chapter 3) where the authors discuss the architecture of their solution along with encountered problems during the development. The content of the mentioned chapter discusses several submodules that form a core of the presented solution. In the beginning, a system for the data gathering is described (further called the Data Collection module; Sections 3.1 – 3.4), which was shared for the public usage during the research conduction. The module

[*] The complete list of links can be found in the Appendix section

architecture incorporates two applications — backend API and frontend application — they were deployed in the cloud and allowed to collect the data for the machine learning model. The next module is the so-called Custom Bot module (Section 3.5), wherein the authors developed a bot that impersonates human-like users basing on the mathematical concept of Bézier curves. The last submodule — the Bot Detection (Sections 3.6 – 3.10) — is composed of the machine learning model itself and a set of tools that facilitate work with a remote computational architecture where the model was trained and stored. Among the mentioned tools, one can find serializer and deserializer tools that allow for effortless data preparation for usage in a machine learning Python script, as well as many statistics, preprocessing, and notification tools.

Following the implementation description, the authors present the results (Section 4.1) of an adapted machine learning approach based on the data collected during the research conduction. The chapter also contains the limitations and conditions (Section 4.2) of the conducted research and the impact of them on the presented results. The thesis ends with the conclusion along with the description of possible further study (Chapter 5) which in the authors' opinion may improve the obtained results.

1.2. Related works

The idea of using behavioral biometrics, especially the one based on mouse dynamics, is still a relatively new concept. The more advanced papers on using the authentication systems that use a mouse movement were created in the late 2000s. In 2007, Awad and Traore presented work [2], in which they introduced a new form of behavioral biometrics based on mouse dynamics that can be used in different security applications. The method is based on neural networks. Later on, they continued their work and in 2009 presented paper [3] that showed promising results. However, in those works the ratios of FAR (False Acceptance Rate) and FRR (False Rejection Rate) were reaching the values above the accepted commercial standards. Therefore, the systems were still not applicable for real-world scenarios.

With the recent technology improvements and standards, the detecting architectures were getting more and more accurate and robust, hence interesting to perform research on them. The works like [14] use deep learning techniques, such as convolutional neural networks, to achieve accuracy over the 96%. The authors gather the data with a custom site built for such a purpose. However, they do not share details for reproducing their system. In another work, Wang Kaixin et al. [6] use a support vector machine classifier to determine the identity of the user based on statistical and dynamical data taken from a low count of users. The process of collecting data is unknown, as the lack of description also remains between different works related to the authentication based on the mouse dynamics.

The remedy for the lacking data gathering information would be to get access to publicly available, verified vast datasets of good quality. One of the best available datasets is the Balabit dataset[4]. The mentioned dataset is used by Antal et al. [1], whose work shows the good quality of the data. However, the given dataset has a couple of drawbacks, like short test sessions and overall lack of data abundance. Chong et al. [4] emphasize the lack of good data source as well and point at a Balabit dataset as the most adequate for the day. The chosen 2D-CNN (Convolutional Neural Network) approach shows interesting and promising results when evaluated on this dataset. To come across the lack of datasets, we propose the architecture of the system for gathering the mouse data from users. Furthermore, the user data along with the simulated bot actions are evaluated on the prepared model to show a potential and suitability for usage in a real machine learning model.

1.3. Contribution

This thesis text and the related system implementation for data gathering, bot, and machine learning model for attack detection was prepared by **Kamil Kaliś** and **Piotr Kuglin**. Below the contribution of both authors is listed and grouped by the module.

Data Collection[9] module:

- The frontend, presentation layer with the Web template, proxy server and its API for login, signup, token obtaining and other utilities, its containerization with Docker along with Heroku deployment, and GCP deployment were prepared by Kamil Kaliś.
- The backend API and OAuth2 server with the database structure, containerization with Docker, performance tests for Heroku deployment, login/signup page layout, and GCP deployment with Kubernetes were prepared by Piotr Kuglin.

Custom Bot[8] module:

- The various scenarios for bot and its execution on the data gathering system and the movement action were prepared by Kamil Kaliś.
- The module for mouse action recording, scenarios execution module and the click and scroll actions were prepared by Piotr Kuglin.

Bot Detection[5] module:

- The deserializing tool, considered previously Avro schema, sbatch input script for SLURM, machine learning execution strategy, methods for collecting execution results and statistics, Imgur integration and machine learning model parameters tuning and results analyze were prepared by Kamil Kaliś.
- The collected data serializing tool, Protobuf schema, data preprocessing, Slack notification integration, scripts for remote job execution on Prometheus with setup automization with *Makefile* and machine learning model parameters tuning with data augmentation and results analyze were prepared by Piotr Kuglin.

Thesis text:

- The related works (1.2), contribution (1.3), behavioral biometrics (2.1) and Bézier curves (2.2) theory, implementation description of: presentation layer and reverse proxy (3.2), custom bot (3.5), Prometheus cluster computation (3.7), Bot Detection module (3.9), machine learning model (partially 3.10) and execution strategy, limitations, conditions and problems (4.2), further work (5.2), and bibliography were prepared by Kamil Kaliś.

- The abstract with work concept (1.1), machine learning theory (2.3), implementation description of: overall system structure (3), persistence and security for backend API (3.1), performance tests (3.3), cloud deployment and orchestration (3.4), data serialization (3.6), data preprocessing (3.8), machine learning model (partially 3.10), results (4.1), summary (5.1) appendix, and glossary were prepared by Piotr Kuglin.

2. Theory

The research was conducted by using knowledge from a couple of domains. The work uses Web and cloud technologies that allow the data gathering environment to be hosted, but they are out of the scope of theoretical introduction. These issues are important but are not in the area of the main concept for this work. More interesting topics included in the work are the issues from the domain of behavioral biometrics, machine learning and the practical usage of the Bézier curves. These are described in the following sections.

2.1. Behavioral biometrics

With the development of more and more powerful technology, there comes a need for sophisticated authentication methods. One of a kind for authenticating users with their individual physical patterns is static biometrics. This approach is widely used nowadays in computers and phones as a way for authentication besides the traditional methods like PIN verification or pattern matching. The main concepts for this kind of biometrics are facial recognition and iris or fingerprint scanning. However, these static methods raise big concerns — when compromised, there is no way to change it dynamically, because data is bounded physically and thus cannot be changed. Secondly, the big progress in technology, especially computer vision and cameras, raise a thread for facial recognition method, where the human face can be simply replicated based on a properly scanned subject head. Moreover, this type of recognition raises concerns about human rights and privacy. The IBM CEO, Arvinid Krishna in his letter [8] to the US government deputies commits the withdrawal of facial recognition in the IBM technology stack.

The more sophisticated, private, and cost-effective implementation seems to be a behavioral biometric technique. Behavioral biometric methods are defined as "any quantifiable actions of a person. Such actions may not be unique to the person and may take a different amount of time to be exhibited by different individuals" [16]. As stated by Tony Thomas in [13] — these actions include voiceprints, signatures, typing patterns, keystroke patterns, or gait. The advantage of such an approach is the fact, that data collecting for further authentication can be gathered in a seamless manner, without the user's knowledge of the process. Thanks to that, the properly designed system may be used to provide a continuous authorization of a user. The malicious, hijacked session would be discovered because, as stated by Yampolskiy: "one of the defining characteristics of a behavioral biometric is the incorporation of the time dimension as a part of the behavioral signature" [16]. Therefore, the data is gathered and analyzed dynamically. The domain can be divided into a couple of main categories, in which there exist the HCI (human-computer interaction) based biometrics alongside the category based on authorship, indirect HCI, motor-skills or direct behavioral biometrics (purely individual behavioral traits, like the way an individual walk) [16]. In the HCI-based approach, the data from different external devices is gathered along with its usage and then analyzed. Among the others, the keyboard keystrokes analytics or mouse pointer usage can be highlighted; the latter seems to be the least discovered method to date. Kasprowski et al. [7] describe well how the data can be gathered and grouped and specifies the variation of experimental environments. Mouse based HCI method can utilize the raw data recorded from the low-level mouse events like the cursor movement or button actions. Those normally include the timestamps and therefore could be further grouped into more high-level actions like drag-and-drop, or move-and-click actions. Data collection can take place in different experimental conditions, like Web browsing, text editing, dot trait following, or game scenarios. Some of the experiments completely discard pre-prepared environments for data collection.

The mouse behavioral biometrics can find its usage in the Web crawlers detection, as Wei et al. [14] raise a topic of huge malicious bot traffic over the Internet. According to Bot Traffic Report 2016 [17], this kind of bot activity reaches almost 30% share of overall Web bots traffic. The mouse behavioral biometrics would fit into the problem, as Wei et al. says — "although web bots can generate mouse events, it's difficult for bots to perform mouse operations in a human manner" [14], so the malicious bot and legitimate user should be classified properly.

2.2. Bézier curves

To achieve a smooth, parametric curve, widely used in computer graphics, animation and font creation, the Bézier curves are used. They are named after Pierre Etienne Bézier (1910-1999) — a French engineer and mathematician with merits in the field of mechanical engineering.

The Bézier curves are based on the *Berenstein polynomial basis*, which was "introduced 100 years ago as a means to constructively prove the ability of polynomials to approximate any continuous function, to any desired accuracy, over a prescribed interval" [5]. In other words, this is the constructive proof of the *Weierstrass theorem* (*approximation theorem*), which "states that the continuous function $f(x)$ on an interval $[a, b]$ and a tolerance $\epsilon > 0$, a polynomial $p_n(x)$ of sufficiently high degree n exists, such that:

$$| f(x) - p_n(x) | \leq \epsilon, \quad \forall x \in [a, b] \tag{2.1}$$

In other words, polynomials can uniformly approximate any function that is merely continuous over a closed interval" [5]. If so, some of the properties for the Bézier curves are derived immediately, for example, it is known that the basis functions are real, the curve generally follows the shape of the control polygon, the first and last points on the curve are coincident with the first and last points of the control polygon, and so on [11].

The Bézier curve of $n + 1$ control points are defined in [5] as:

$$r(t) = \sum_{k=0}^{n} p_k b_k^n(t), \quad t \in [0, 1] \tag{2.2}$$

Where the $p_0, \dots, p_n$ are the *control points* and b_k^n is the Berenstein polynomial:

$$b_k^n(t) = \binom{n}{k} t^k (1-t)^{n-k}, \quad \binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{2.3}$$

For example, the two-dimensional Bézier curve is described as a pair of a polynomials:

$$(x, y) = \left(\sum_{k=0}^{n} x_k b_k^n(t), \sum_{k=0}^{n} y_k b_k^n(t) \right) \tag{2.4}$$

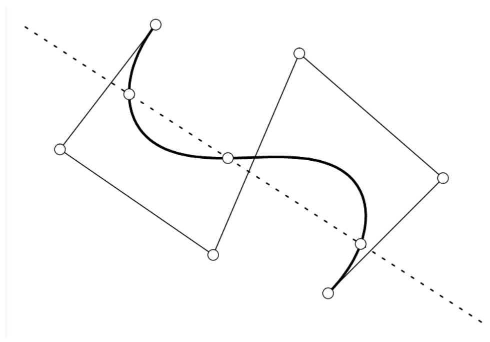


Fig. 2.1. Example of Bézier curve fitting to the control points from [5]

The control points $p_0, \dots, p_n$ shape the control polygon for the Bézier curve, where the first and the last points on the Bézier curve are coincident with the control polygon (detailed proof and mathematical derivation can be found in [5]). The properties of the Bézier curve allow for creating the smooth shape, that is going through a set of predefined points and the curve is trying to 'follow' the polygon. This makes this solution convenient from the point of view of this work because the prepared bot is expected to make moves in a human-like manner, so the moves should be smooth and look natural.An example of the control polygon with the Bézier curve shaped by it can be seen in Fig. 2.1 from [5].

2.3. Machine learning techniques

In modern cyber security, there exist many different threats that potentially expose the system to being compromised. Many of them are very sophisticated, some are pretty difficult to distinguish from legitimate operations. Nowadays, there is a trend to use machine learning approaches to meet the high demands for the quality and reliability of security systems.

As stated by Tony Thomas et al. — "Machine learning (ML) may be defined as the ability of machines to learn without being explicitly programmed. Using mathematical techniques across cyberdata, ML algorithms can build models of behaviors and use those models as a basis for making predictions on newly input data" [13]. This behavior of machine learning techniques is very convenient, especially when the domain of the problem and the borders between the data cannot be explicitly expressed by the written program. Machine learning models can learn from their own mistakes and then recognize or even predict the future attacks [13].

In scope of sequential data with natural timespan interpretation such us mouse actions, the recursive neural networks are natural choice. As stated by Chong et al. — "Given the sequential nature of mouse movement data, a recurrent neural network (RNN), commonly employed for time series structure data, would be an intuitive choice for tackling this problem" [4]. However, if the data can be represented as a bit map or a picture, the convolutional neural networks become possible to use. Due to the size of picture representation in a computer world, it is pretty expensive in the meaning of computational cost to use plain artificial neural networks because the number of weights in such a network becomes tremendous. Convolutional neural networks address this issue and make the computation significantly faster and efficient.

Keiron O'Shea et. al in [9] define an architecture of CNN's as a connected network of three types layers: convolutional layer, pooling layer and fully-connected layer. The convolutional layer is combined out of filters that represent features of the images that the given filter should recognize. These filters are then used on the different regions of images from the dataset by performing convolution. The output reflects the found matches between the image that is recognized and the one described by the filter. Pooling is a technique used to decrease the size of the image by grouping pixels and returning the representative value for this group. As an example, the max-pooling bases on returning the pixel with the highest value in the group. The fully-connected layer is a layer that builds plain artificial neural networks and it is built out of the neurons that are connected to the neurones from the previous layer, but they are not connected inside the current layer. Thomas et. al add to these layers so-called rectified linear unit layer (ReLU) and define its responsibility as "changing the negative pixel values in the image to zero, which gives us another stack of images with no negative values" [13]. This layer is called an activation function because it activates the next layer only if the value of the pixel is positive.

The problem that this work raises can be specified as a binary problem because there exist only two possible categories for the data — user's and bot actions. In such situations, the measure of quality and correctness of the solution is commonly defined by a confusion matrix. This matrix defines the performance of the model and consists of several measures: true positives, true negatives, false positives and false negatives. True positives (TP) are defined as the number of samples that the model assign to the positive category and the assignment is correct. The opposite to them are false positives (FP), which can be described as faulty categorized to the positive category. The true negatives (TN) and false negatives (FN) are analogous, but the assigned category is negative.

Basing on the described measures, one can define relative measures — false rejection rate (FRR) and false acceptance rate (FAR). These are defined as follows:

$$FRR = \frac{FN}{FN + TP} \tag{2.5}$$

$$FAR = \frac{FP}{FP + TN} \tag{2.6}$$

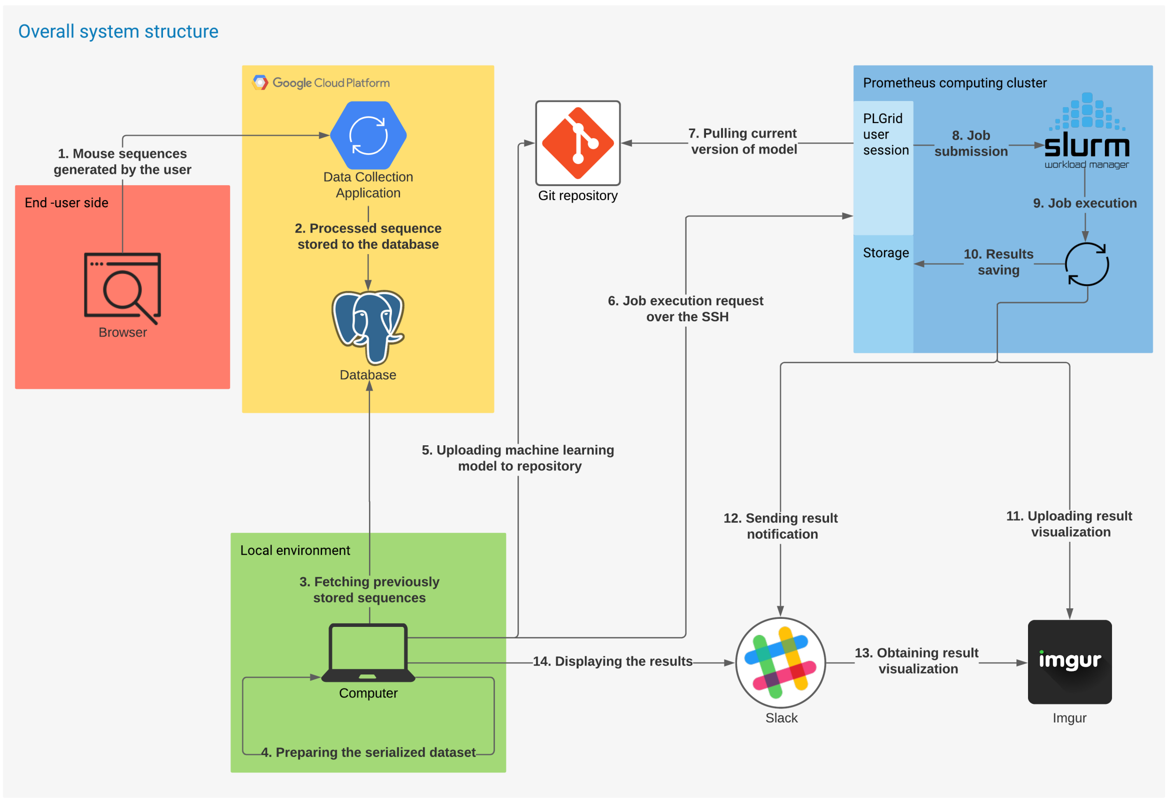
The values of FRR and FAR are often expressed as a percentage value and the smaller their values, the better the performance of the model. In authorization related problems such as raised in this work, there is a desire to have those values as low as possible because low FAR defines the good security level of the system — the lower it is, the less unauthorized users have access to the system, and accordingly FRR has an impact on authorization rejections of legitimate users.

In the presented solution, the authors use a transfer learning technique. As stated in the "A survey of transfer learning" by Weiss et al. — "In certain scenarios, obtaining training data that matches the feature space and predicted data distribution characteristics of the test data can be difficult and expensive. Therefore, there is a need to create a high-performance learner for a target domain trained from a related source domain. This is the motivation for transfer learning. Transfer learning is used to improve a learner from one domain by transferring information from a related domain" [15]. This approach provides several conveniences, such as computational time and cost reduction. The pretrained network does not require such a long time as in the case of training from scratch which also translates into the cost of calculations and in the end does not require a large computational grant. The main advantage of such an approach is the reduction of the required amount of representative data. In cases where the data gathering is difficult or the collected dataset is small, transfer learning seems to be a technique that is worth considering. As a transfer learning model, the authors chose the InceptionV3[17] architecture from Tensorflow Hub[32], which originally was trained on ImageNet[15] dataset. Basing on several benchmarks, it was considered that the described architecture fits well to the raised problem and the performance is great at the same time. The architecture is build out of convolutional layers, average and max pooling, dropouts, concatenation layers and fully-connected layers. Description of the InceptionV3 architecture is out of the scope of this work.

In order to perform the learning process of the model, the main dataset should be divided into two subsets — training and test datasets. The network basing on the output of the loss function is able to improve its predictions and make progress in classification. In the case of the supervised learning, where the network learns from the given examples, there is also a requirement for the assignment of the labels that are considered as descriptive ones for the input sample. When the gathered data consist of several categories, among which there is a dominant class in the meaning of volume of samples, such dataset is considered as an imbalanced one. This issue has a negative impact on the learning process and should be resolved in order to improve the performance of the model.

3. Implementation

To fulfill the assumptions of this work, the authors prepared a system that is able to collect and verify the mouse dynamics data. The high-level flow of the system is presented in Fig. 3.1 and provides the basis for further considerations.



Overall system structure
Google Cloud Platform
Data Collection Application
1. Mouse sequences generated by the user
End -user side
Browser
2. Processed sequence stored to the database
Database
Git repository
7. Pulling current version of model
Prometheus computing cluster
PLGrid user session
8. Job submission
slurm workload manager
9. Job execution
Storage
10. Results saving
6. Job execution request over the SSH
5. Uploading machine learning model to repository
12. Sending result notification
11. Uploading result visualization
Local environment
3. Fetching previously stored sequences
Computer
4. Preparing the serialized dataset
14. Displaying the results
Slack
13. Obtaining result visualization
imgur
Imgur

Fig. 3.1. Overall system structure diagram

The entry point for the system is the user's browser that allows the human user as well as a human impersonating bot for access to the prepared website which is hosted on the Internet. The Data Collection[9] module which is persisted and operates within a cloud acts as a mouse dynamics data collector, which means that every single mouse event generated by the user on the website is intercepted, transformed and stored in the underlying database (Fig. 3.1, pt. 1 and pt. 2). The administrator of the presented system is able to retrieve the data from the database in any time (pt. 3). The downloaded data can be further processed (pt. 4) to the dataset which will be used in machine learning stage. The administrator of the system should prepare a machine learning model and upload it to the Git repository (pt. 5) which will be then obtained by a computing cluster to perform computation using the model from the observed branch. Such an approach allows to work on the solution simultaneously by many data scientists and makes it possible to accelerate the research. The dataset should be uploaded to the computing cluster and persisted in the group's storage that allows using it by many different paralleled computations (not included in the diagram due to decrease of readability). Each computation is requested from the local computer using prepared Git's 'deploy' alias (described in Section 3.7) that performs a sequence of operations such as establishing the connection to the cluster, fetching the current version of code from the repository and submitting the job to the SLURM[28] workload manager (pt. 6, 7, 8). The work of the cluster is fully asynchronous because each job is queued and therefore the completion time is unknown. This is very inconvenient because there is no notification system that allows getting the information about the finished job. To overcome these limitations, the notification system is proposed as a part of Bot Detection[5] module. The creation of such a system enables the presentation of the results of the computed job in the message as well as the graphs prepared based on the output of the job. To provide the possibility of sending the images, the notification system uses the external image hosting website called Imgur[16] which allows to upload pictures to the server and host them under the generated URL (pt. 11). The results and the graphs are combined into Slack's[29] message and then sent to the previously prepared channel by using the webhook API (pt. 12, 13).

The further subchapters treat about the implementation details. At the beginning the Data Collection module is described along with the cloud configuration and performance tests, further, the bot which impersonates the human user and finally the machine learning model alongside the tools such as, among others, a notification module.

3.1. Persistence and security

The Data Collection module consists of several submodules, but the core of it is the backend API which was written in the 8th version of Java. The responsibilities of that API are mainly the persistence of the captured data and user authentication. The whole architecture of the API is based on the Spring[30] framework which is widely used for such purposes and provides many capabilities in fields of securing and persisting data. Spring is divided into many projects that create compatible ecosystem and make it possible to develop scalable Web applications. In order to make development in Spring easier and faster, the authors of the framework created Spring Boot[31] which sets up the environment for the developer and provides many easily imported libraries (so-called starters) to use.

To perform database operations using the idea of ORM (Object–Relational Mapping), the implementation of JPA (Java Persistence API)[19] provided by Spring Boot was used. The choice of the database was in the scope of relational databases because the captured data forms the time series which are easily mapped to the structure of the table. The decision was made to use PostgreSQL because of the popularity and wide range of support. The data was stored in three different tables. The first one consists of the user's login, password, and granted authority which was required to distinguish administrators from ordinary users. The second one includes sequences of user's captured actions with timestamps, screen resolutions and types of the events. The last one stores all possible action types with the description.

The sequence in this work is treated as a chain of actions that were performed by a user or bot, one by one. Each element of the sequence has associated several properties such as coordinates, timestamp, and action type. The sequence is distinguished, one from the other, by a so-called split point which is established by the time that passed between two actions. If the time gap between two consecutive actions is greater than the given threshold the former action is considered as the end of one sequence and the latter one as the beginning of the second. The sequence length is the number of consecutive actions in one sequence and it depends on the split point.

The collected data is sent to the API using REST (Representational State Transfer), which is nowadays one of the most popular architectural style for building interfaces between Web applications, because of its simplicity and native support in many different programming languages and technologies. The schema of the interface was established at the beginning of the work on this module, so it was possible to create the API and the frontend application simultaneously.

To protect the data from invalid properties like negative coordinates or invalid timestamps, the input was also validated before saving to the database in such a way that all of the persisted data contained required elements. Securing such an application from malware users is also an important issue, so the access was restricted only for signed users that existed in the database. The security was provided by using OAuth2[22] and JWT (JSON Web Token)[18]. The API was working as an OAuth2 provider which issued JWT tokens and authorized them basing on the user's credentials stored in the database. Such a mechanism allowed to control access to the application and enforce registration before the use of the system.

The API was prepared with two different switchable profiles: admin oriented and user oriented. They were created to provide flexibility in accounts creation — respectively — only by admin and anyone. In further work, only user oriented profile was used.

3.2. Presentation and reverse proxy

The second core submodule created for the purpose of collecting the data from the users is the frontend application exposed to the Internet. This can be split further into two different functional modules — client-side and server-side.

3.2.1. Client-side module

The client-side module is an end-user presentation layer that is built with the HTML5 and CSS3 — technologies that are the core and standard for modern Web applications nowadays. HTML is a widely used markup language used to create hypertext documents and CSS is used as a style-sheet for modifying the design of Web documents. The content of the website is a free, prebuilt template taken from the Colorlib[6] collection which Web templates are licensed under the CC BY 3.0 License[7]. The mechanism that collects users' mouse actions is designed as a 'plugin' script, meaning that the template can be changed easily, so different styled environments that serve various purposes can be used to collect the data.

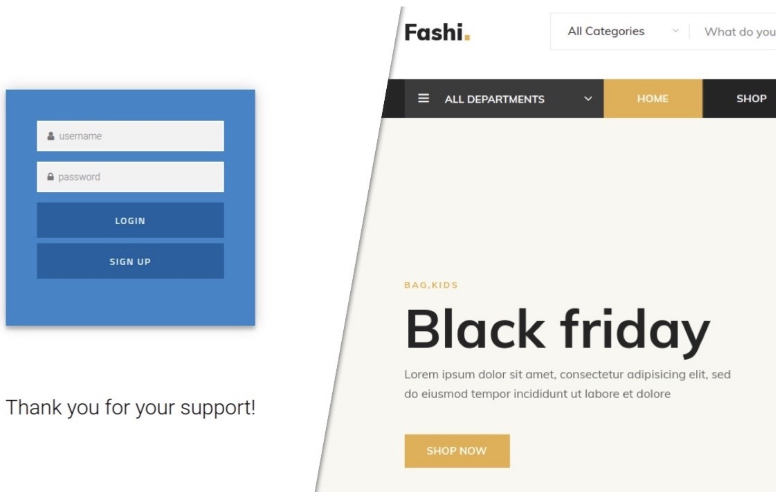

username
password
LOGIN
SIGN UP
Thank you for your support!
Fashi.
All Categories
What do you
ALL DEPARTMENTS
HOME
SHOP
BAG,KIDS
Black friday
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore
SHOP NOW

Fig. 3.2. Login page and landing page layout

To use the system and participate in the research, an interested user is required to first accept the consents of usage of the website as well as accept the usage of the cookies and then registration is allowed and required. Afterward, the user is redirected to the login page. To persists consistency of the registration and login, the custom static Web pages were created to be easily transferable between the templates. On each document, the FAQ panel with more information regarding the project is exposed as a drop-down list. The layout of the login page, and landing index page is shown in Fig. 3.2. To authenticate, users are required to provide the credentials to log in to the system. These credentials are then sent through HTTPS — which means that they are secured and encrypted — to the reverse proxy server which then performs actions to authenticate a user, sets the cookie with granted JWT token and redirects the user to the homepage. More on the user authentication and the whole token obtaining sequence is described and shown in Subsection 3.2.2.

After a successful login, the token allows for site usage and ensures the identity of the user. From now on, the actions performed by the user are recorded and grouped into the batches and sent to the API every two seconds. The event listeners are awaiting four different action types: mouse move, mouse-down, mouse-up, mouse wheel action. Collected actions are packed into JSON object and sent to the server side API (described in Section 3.2.2) — the fields included are shown in the Listing 3.1. The schema holds the list of the mouse event objects and it is called *mouseEvents*. The single mouse event object has the *x_cor* and *y_cor* keys with the values of the respective x and y coordinates of the mouse pointer on the screen. The values *x_res* and *y_res* represent the resolution of the screen. To distinguish the type of the mouse action and place it on the time axis, there are *event* and *event_time* attributes. The event time is saved in the datetime format of *'yyyy-MM-dd HH:mm:ss.SSS'* (example: 2021-01-25 12:23:31.334). Event has following predefined values: *MOVE*, *LEFT_DOWN*, *LEFT_UP*, *RIGHT_DOWN*, *RIGHT_UP*, *SCROLL_PUSH*, *SCROLL_PULL*, *SCROLL_DOWN*, *SCROLL_UP*. They respective meaning is: mouse movement, left button pressed, left button released, right button pressed, right button released, wheel pushed, wheel released, wheel-scroll down, wheel-scroll up.

```json
{
  "mouseEvents": [
    {
      "x_cor": "int",
      "y_cor": "int",
      "event": "string",
      "event_time": "yyyy-MM-dd HH:mm:ss.SSS",
      "x_res": "int",
      "y_res": "int"
    }
  ]
}
```

Listing 3.1. JSON schema for mouse event batches

3.2.2. Server-side module

The server side module is built with Node.js[21] runtime with the Express[11] framework on the top of it. Node.js is an asynchronous JavaScript runtime, which is widely used to build high-end, scalable commercial applications. Express is simple to use, yet powerful Web framework for Node.js, which allows for a fast and convenient HTTP server set-up for serving the static Web documents over the Internet. This module is serving the purpose of reverse proxy between clients and the backend API.

The main responsibilities of the reverse proxy are signing up the user, token granting, validation and caching, serving static Web documents storing and passing the data for persistence. When a new user tries to sign-up, secured with TLS (Transport Layer Security) credentials from the sign-up form are sent to one of the proxy endpoints, where they are then passed to the backend API. Since the backend is not exposed anywhere over the public network, there is no need to secure the credentials with TLS. The user is granted JWT Access and Refresh tokens after logging in using the login form. The user credentials are secured with TLS and received by proxy API, whereas the proxy server is additionally appending the Client id and Client secret for OAuth2 server to the request 'Authorization' header, as the proxy server authenticates to OAuth2 server with HTTP Basic authentication scheme.

Credentials of the user who wishes to log in are included in the body of the request. When the user is properly authenticated, the OAuth2 server responds with a valid JWT token which is then set as a cookie with HttpOnly, Secure, and SameSite strict options. When the token is successfully granted, it is also saved to the Redis, which is a simple to use key-value NoSQL database suitable for caching. The retrieval of such a token is very fast, so this is serving the purpose of the caching system, which results in a great efficiency improvement and it lowers the response time of the server. For each interaction and request for resources, the user has to hold a valid JWT Token. First, the token is searched in the cache database. If it does not exist there, the proxy server is attempting to check the token with the backend OAuth2 server. If the server responds with Bad Request status or the access token is not set on the user request, the refresh token is being used for access token renewal. The complete sequence visualization can be seen in Fig. 3.3. The proxy server is storing the user mouse data received from the client side and periodically passes it to the backend API for persistence.

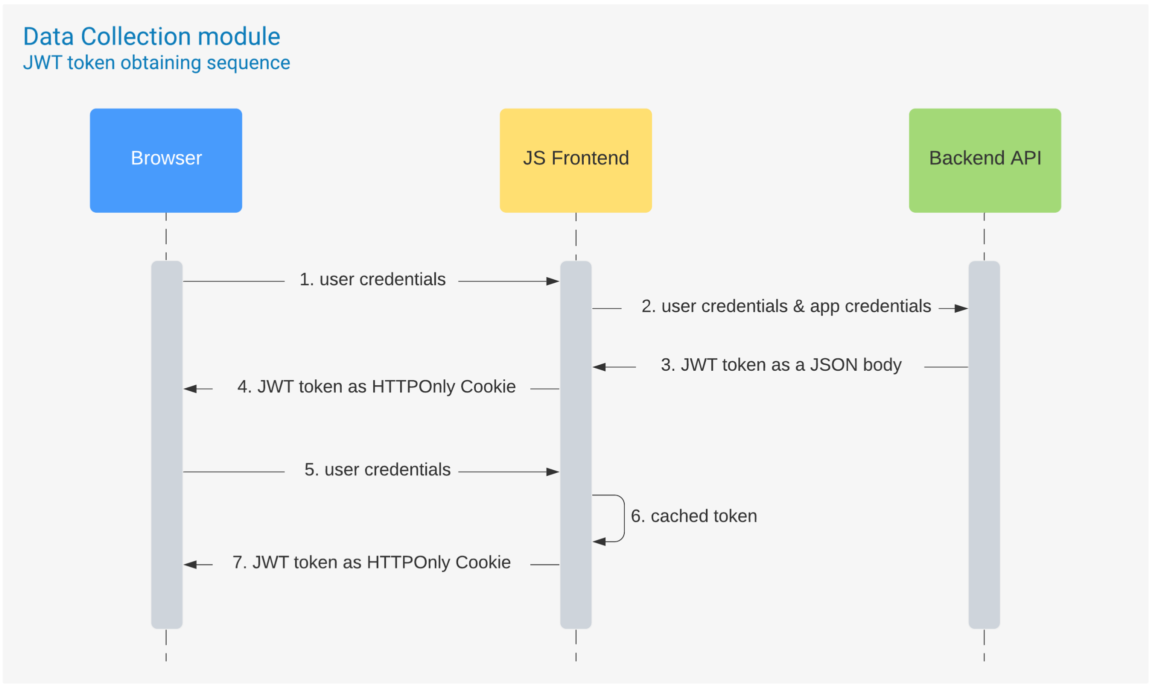

Data Collection module
JWT token obtaining sequence
Browser
JS Frontend
Backend API
1. user credentials
2. user credentials & app credentials
3. JWT token as a JSON body
4. JWT token as HTTPOnly Cookie
5. user credentials
6. cached token
7. JWT token as HTTPOnly Cookie

Fig. 3.3. JWT token obtaining sequence

3.3. Performance tests

The solutions presented in the previous sections were deployed to the Heroku Cloud Application Platform[14] to deliver the possibility of using the system among a wide range of users. Heroku as a cloud provider has a free pricing plan for non-commercial apps, which is suitable for scientific usage such as this thesis. The deployment process is performed using the idea of containerization based on Docker[10] which is a convenient way of application delivery to the Web server. To enable such a possibility, the Dockerfiles with container configuration description were prepared for both frontend and backend services and also docker-compose configuration files were created to allow easy testing and development in the local environment.

In order to deliver reliable and efficient product to the users, the performance tests were prepared using JavaScript programming language in the Node.js environment. Those tests cover mainly the token exchange and refreshment, and also allow to test the behavior of application during increased load produced by using the system by many different users. The results of the tests showed that the Heroku free pricing plan had some limitations regarding database storage volume and response time from the application. It turned out that the database in such a plan permit to store only 20,000 rows overall which is unacceptable for the data stream with the time resolution measured in milliseconds because a single user may cause exceeding the limit in just a few minutes, which disqualifies the Heroku as a service provider for a multi-user environment. Moreover, the described cloud system suspends the work of the application when it is idle. Waking the application up takes a very long time (up to the minute or more), which is manifested in a usage lag after 30 minutes without any Web traffic. Described issues with the Heroku Cloud Application Platform forced the authors to find another suitable cloud provider.

3.4. Deployment and orchestration

In order to overcome the issues described in Section 3.3, the cloud providers' research was conducted. Basing on the results, it was decided to adopt the GCP (Google Cloud Platform)[13] as a deployment environment. In the scope of the conducted research, some popular cloud providers were considered, such as GCP and AWS (Amazon Web Services)[1]. Due to the similar costs of both options, the choice was directed by the prior experience that authors had with the GCP.

Unlike Heroku, GCP had no free pricing plan, but it does not restrict the usage of resources and therefore it met the project assumptions, and moreover, it has some extra advantages like native support for containers orchestration. The latter is provided by using Kubernetes[20], the open-source solution introducing flexible deployments, scaling and container management. Those features drastically simplify working with application development in a cloud, but require additional setup and configuration files.

Kubernetes natively supports the usage of Docker containers, so the ones prepared before can be reused. However, the deployment process requires a so-called Deployment configuration files that enable tuning the resources assigned to the single instance of an application and also the number of the replicas of the application. The basic unit managed by Kubernetes is Pod which may consist of many single Docker containers, but in this work the Pod is associated with a single container of application. The described scaling method is known as Horizontal Pod Scaling and can be seen in Fig. 3.4 as a several same Pods in a single Deployment. It improves reliability and allows to increase the limit of the maximal load accepted by a single deployment of the application because the load is split into the mirror instances of the same application that work in parallel. In Fig. 3.4 the Deployments are drawn as rectangles which states that all of their content is in some measure related to them.

To deliver the load to all of the mirror applications, Kubernetes uses Services as an entry point to a group of Pods that are managed by one Deployment (in Fig. 3.4 they are drawn on the edge of the Deployment rectangles to mark their gateway nature). The Service exposes the Deployment under the

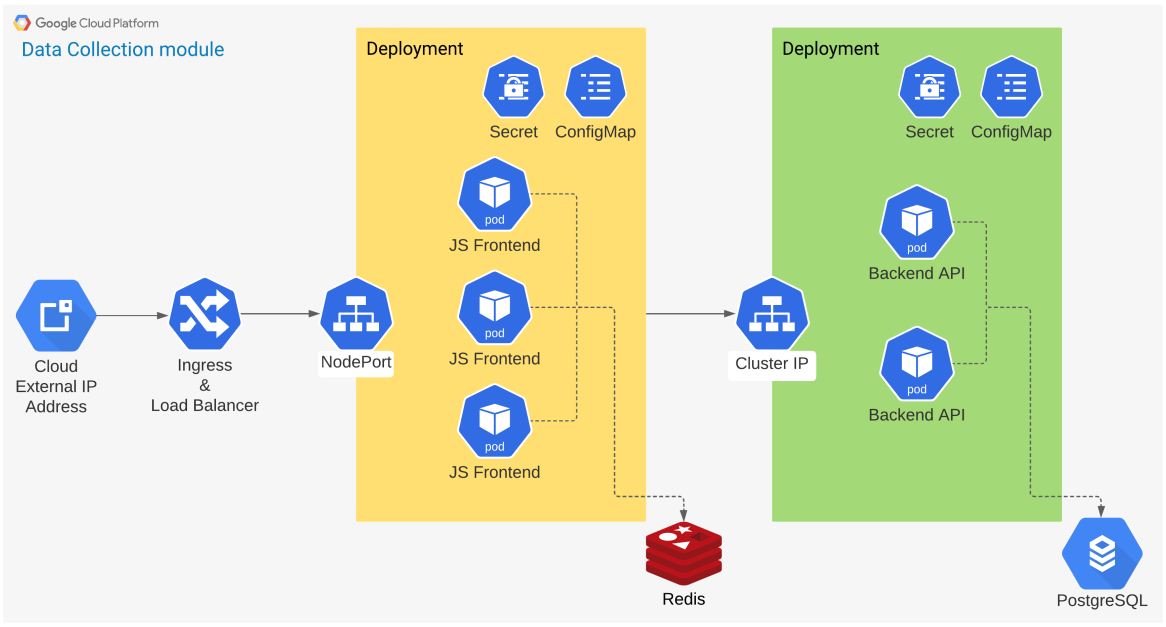


Fig. 3.4. Cloud architecture

single DNS name and updates the underlying IP address in cases of its change, but it also works as a load balancer which distributes the load among the managed Pods. Kubernetes has several different Service types, but in this work two of them are used: Cluster IP that exposes the Deployments in the scope of Kubernetes, but hides them from external access, and NodePort that permits the access using port-forwarding. The latter one is dictated by the usage of Google Cloud HTTP(S) load balancer and it also requires additional configuration by using an Ingress and obtaining the static external IP address.

To provide encryption of connection, TLS should be used, which requires the SSL Server Certificate, which can be issued only for existing valid domain names, thus the authors were forced to buy such a domain name, while the certificate is issued by GCP.

The configuration also includes some sensitive data, such as database credentials, OAuth2 secret, and SSH-RSA keys, thus, the Secrets are used with encoded content. The insensitive data are included in ConfigMaps that are similar to the Secrets, but their content is a plain text. These configurations are matched to the appropriate component by labels and therefore they are drawn inside the Deployment rectangles in Fig. 3.4. In practice, they can be mounted as volumes basing on the component labels and can also expose their values as environmental variables.

The whole architecture of the collector system is presented in Fig. 3.4, but the figure applies several simplifications such as a way of database deployment. In contrast to the Redis cache database, the main database instance is not deployed in Kubernetes itself, but the Cloud SQL from GCP available services with PostgreSQL is used to achieve straightforward management and reliability.

The flow of the presented architecture starts with the external global public IP address that is also able to be resolved by using DNS. The request from the user's browser is transitioned to the Google Cloud

HTTP(S) load balancer that is able to deliver the load to different nodes of the cluster even in different regions and zones. In this work, due to the limitation of costs, the Kubernetes cluster was limited to one region and one zone so the usage of load balancer relied mainly on passing the load from the public IP address to the Kubernetes. In the next step, the load goes to the Deployment of the frontend application through the NodePort service where is divided between several instances of the same application. This division allows for seamless changes of application versions and also decreases the requirements for efficiency of a single application instance. Some of the requests such as authorization ones can have their data cached in the Redis database, but most of them are passed to the backend API in the second Deployment. As in the previous case, the entry point for this Deployment is Service (Cluster IP), which also serves the load between some instances of the backend API. These applications use the underlying database to persist their data, such as user credentials and collected mouse events. The response for the request is at the end passed back through all of these layers to the end-user and displayed in his browser.

3.5. Custom bot

In order to prepare a dataset that consists of two classes — the valid user and human impersonating bot, the script that was executing the human-like actions was prepared.

The main idea behind this module is to first create scenarios that next could be executed by the bot software. To achieve the most convenient creation of scenarios, the authors prepared a utility to record the coordinates and actions of the mouse. Using this approach, the bot could impersonate a user in a more natural way. Recorded mouse events are stored as a YAML schema, which is a human-readable standard for data serialization. The structure of a YAML scenario schema is shown in Listing 3.2.

```yaml
id: "string"     # Scenario's ID
name: "string"   # Scenario's name
sequence:    # List of element objects
    - element:
          delay: "numeric"     # delay between each event
          dx: "int"    # offset generated by mouse wheel movement in x axis
          dy: "int"    # offset generated by mouse wheel movement in y axis
          event: "string"     # mouse event that triggered the action
          ord: "int"   # event order
          x: "int"     # mouse pointer coordinates in x axis
          y: "int"     # mouse pointer coordinates in y axis
sequence_length: "int"   # count of mouse events
```

Listing 3.2. YAML schema for bot event scenarios

After the scenarios are recorded, the mechanism to execute them is required. The Custom Bot executor script reads the YAML file and loads it to the memory as a list of events, that is then traversed. To be able to run a set of scenarios, the module introduces a so-called ‘batch runner’ for predefined cases. This automation saves a lot of time and effort because a set of scenarios can be executed at once to generate data for the bot class. The executor script performs an action stored as an ‘event’ filed at the coordinates (x, y). If the coordinates are different than the current mouse position, the cursor is moved, but to make it more natural, movement is based on the Bézier curve concept, using the PyClick[26] Python package. The Bézier curves are used to effectively represent a smooth curve on a computer screen, as stated by D. F. Rogers [11]. Knowing this, this concept can be used to simulate smooth human-like movements. The data generated in such a way may be prone to be suggestive for machine learning algorithms that would learn the Bézier curve concept. The verification process focuses on distinguishing between a human and an artificial bot that can be constructed in many different ways. Some bots available on the Internet are even simpler than the presented one — many of them move from point to point in a straight line. If behavioral biometrics is proposed as a potential securing method, the more sophisticated bots such as the ones that use the machine learning techniques will be probably used. However, the way of bot creation has no impact on the data gathering system, so more sophisticated methods are out of the scope of this thesis.

3.6. Data serialization

The persisted data from the database is to serve as the input for a machine learning model. However, due to inconvenient usage of SQL queries for such purposes, the serialization tool was developed in order to translate the state of the database records to binary files. In the presented solution, such files are treated as immutable, so the operation of serialization can be performed only once, which results in the improvement of the required time to read the data by the machine learning model. Moreover, binary files allow straightforward sending and storing data in external infrastructures, such as PLGrid[23], because it does not require maintenance of database engine in that environment. Such an approach also makes it possible to process data and prepare them in a way that is required by the machine learning model.

To fulfill these requirements, the serializer and deserializer tools were developed for saving sequences in binary files and further reading those data in Python script. The serialization is performed using a tool written in Go[12] and provides additional tuning of resulted binary files and configuration of connection to the database. Among the others, the tool allows choosing the type of an event generated by the user, such as mouse click or move, minimum sequence length that should be considered as valid data, minimum screen resolution in order to filter the actions from mobile devices or the time gap between two actions that should be considered as the boundary between two sequences. The results of such filtration are saved in the chosen directory dividing the output into the user’s directories and saving each separate sequence in a single file, so the output consists of many user’s directories each containing many single sequence files. It is also possible to use a so-called one-user mode that enables generating output data only for a single arbitrary chosen user for debugging purposes.

In order to make serialization easy and transferable between different programming languages, the serialization framework was used. At the beginning, Apache Avro[2] was chosen. It allows for defining the schema in a simple JSON file and the serialization is performed with help of the library that allows reading schema and saving programming language native objects to binary files. In the presented solution, the serialization should be performed using the library for Go and the deserialization with support of the library for Python. Unfortunately, they proved to be incompatible which resulted in errors in deserialized data.

To avoid invalid data and to do not spend too much time on finding the bug in those libraries, another approach was taken by applying the Protocol Buffers[25] technology. Protocol Buffers, or simply Protobuf, is a method of serializing data to the binary form, but the real advantage of it is official multilingual support by generating a serializing code in a required programming language. Protobuf also requires the definition of a schema like Apache Avro, but unlike Avro, the config file format is developed especially for Protobuf. At the same time it is also readable and effortless to write. Basing on the created schema, the code was generated both for the serializer and the deserializer, but in the case of deserializer, the data is directly read to the Pandas[24] Dataframe objects, which provides a simple interface to manipulate huge amount of data. The deserializing tool was designed to work directly in the front of the neural network to reconstruct the serialized binary data. Therefore, it was written in Python to provide flexibility in adapting this feature in further work. It is also able to read data from the directory tree created by the serializer which enables a seamless integration between the data gathering system and Bot Detection module[5].

3.7. Prometheus computing cluster

Data processing and machine learning model training and evaluation require a lot of computing resources. Performing such computations locally, on a single processing core, is not the best approach, at least not the most effective method. To enhance the process of manipulating the collected data, the computing cluster was used as a part of the educational grant issued by the PLGrid. To take the full advantage of the computing cluster, plenty of helpers and steering scripts were prepared to automate the process of managing the Prometheus supercomputer. Prometheus cluster is sharing the resources among multiple users. Therefore, the computation is scheduled as a job by the SLURM Workload Manager[28]. To submit a batch job to the SLURM manager, the input shell script is needed and in this case, the *sbatch_job_config* script which contains the job description was prepared. This includes all necessary configuration parameters for the SLURM manager, like job name, time, grant name, nodes count, GPU partition selection, CPUs allocated with the memory, modules added to properly run the main Python script. On the top of the *sbatch* script, a helper shell script exists — *run-plgrid-job*, which is obtaining useful values that steer the SLURM scheduler as well as triggers the notification job described in the paragraph devoted to Slack notifier in the Section 3.9. These notifications are really convenient, because one don't need to constantly check for the job status, but instead for every state, there is an appropriate notification for job schedule, running, crashed, and finished job.

For automation, the Git version control system was utilized to implement a very simple continuous delivery system. For delivering and scheduling the job on remote host, the *git deploy* alias was created and can be set up using the *create-git-alias* script. When alias is created and used, the script named *git-push-alias* is launched which triggers the *git push* command to update the repository and next the routine for executing the job on remote host is triggered with *run-commit-execution* script. After successfully submitting, the user receives a notification on Slack. When the job starts, crashes, or finishes properly, the notification is sent again to the Slack. This gives a very convenient insight into the process and progress of the job without constantly checking for the status manually. The whole setup can be done easily via the *make* program with prepared *Makefile* file. The routine sets all necessary information on local and remote system. User needs to provide the PLGrid username, branch name from repository to observe and execute the model from on the Prometheus cluster. Routine is also generating the SSH RSA key-pair for logging in seamlessly. At the end, the necessary environmental variables are exported.

3.8. Data preprocessing

To fulfill the requirements of the adapted transfer learning model the preprocessing tool was developed alongside a whole bunch of settings. The collected data sequences are in form of coordinates' series, so it is necessary to represent them as 3-dimensional pictures which are the input to the convolutional neural network. The described tool allows transferring those sequences into multidimensional arrays of numbers and also scale them to the required input size of a used transfer learning model.

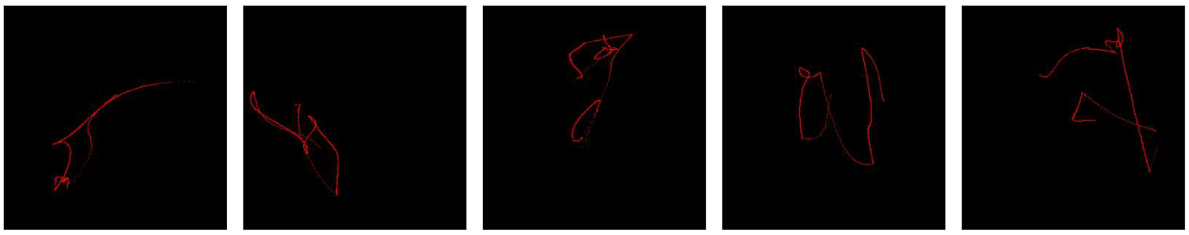


Fig. 3.5. Collected and preprocessed bot sequences with applied linear interpolation

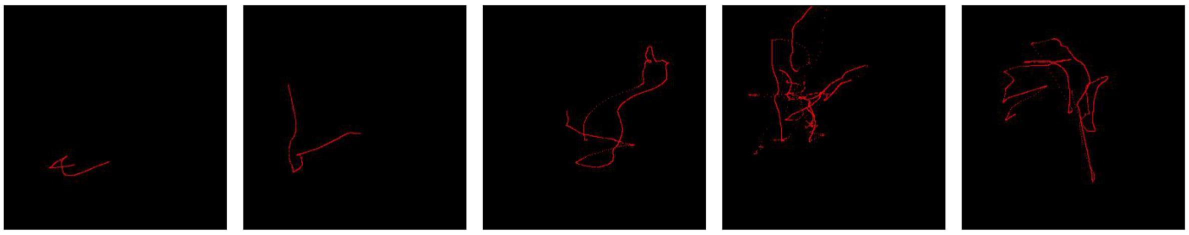


Fig. 3.6. Collected and preprocessed user's sequences with applied linear interpolation

The data consists of isolated points that are result of discrete sampling, so to improve performance efficiency these points that represent the original coordinates on the user's screen are interpolated using a linear interpolation technique. The operation was performed by connecting the discrete points using straight lines, so each consecutive event's coordinates were combined into a chain that results in a continuous curve on the input pictures. The input data with applied linear interpolation and resizing are visualized in Fig. 3.5 and Fig. 3.6. It was verified on the collected dataset that interpolated data results in better accuracy of a model in comparison to the isolated ones. The presented tool works also as a splitter between training and testing sets using the provided ratio between them and as a label assigner which allows adding the corresponding labels to the single sequence basing on the provided identifier of the bot user. Using this tool it is also possible to increase the number of bot samples by using several repetitions of the original bot set in cases of an imbalanced dataset. The output of the preprocessor is a tuple of training and testing dataset along with labels, which can be directly the input of a neural network model.

In further work, in order to improve the performance of the model, data augmentation was introduced and applied. A detailed description will be provided in Section 3.10, but it is worth mentioning that this approach required changes in the preprocessor tool. The preprocessor was extend to allow saving interpolated and scaled images to the directory provided by the user. Along with the extensions in the preprocessor, the main script was enhanced with the ability to load the dataset from image files. The previous option — loading from binary files — was unchanged.

3.9. Bot Detection module

To validate the consistency and quality of the gathered data, the Bot Detection module[5] is introduced. This module includes various tools and utilities helpful for manipulating the data, model observations and results analysis. The selected submodules are briefly described in the following paragraphs.

CSV writer To persist machine learning model execution results and statistics for later analysis, the outcome data should be stored in a consistent way. The CSV file is a convenient file format to persist this kind of data because the values from different execution sessions can be easily appended using established schema. The *CSVWriter* class introduces methods for appending the data to the file. The other helper methods allow for checking the correctness of the file and its content.

Uploading charts on Web To see whether the machine learning model is actually getting better during the training process, accuracy and loss charts can be used as a really helpful method to determine the learning curve. To receive a 'live' preview of the model execution results with charts as a notification in Slack communicator, the charts needed to be stored somewhere on the Internet. The easiest way to achieve this requirement was to use the API of the Imgur[16] platform. Thanks to the official module that can be used with Python, the platform offers a simple way to upload the images to the service and returns the URL to the stored image when successfully uploaded.

Results terminating When the model was executing the calculations multiple times, the results for each iteration were collected. This is giving a really interesting insight into the model and its stability.

The tool called *ResultTerminator* was serving this purpose. The helper methods were used to determine whether the file is existing already and if not creating it. To avoid concurrent saving problems, the file is achieving the lock and releasing it when the content is successfully saved to CSV file.

Slack notifier Working with PLGrid infrastructure and running a job on the Prometheus computing cluster was very helpful in terms of delegating the great amount of computing load onto external resources. However, this solution has its downsides. One of the major issues is the job queueing mechanism, which does not determine when the job will be executed and therefore it requires constant manual checking for results. To avoid that, the authors prepared a custom mechanism, that can monitor the status of the job and notify the Prometheus user through the Slack[29] channel using observability-like architecture. The following message templates were prepared for the purpose of notification:

1. Simple message — Generic message type built as a JSON template with help of Builder design pattern to have flexibility in customizing the message.
2. Pending job message — able to notify about: job title, reporter, commit hash, job registration date and time, header for app preview, info message taken from a commit. It is built as a special case of Simple type message.
3. Results message — The most complex message with all the statistic results produced as a result of the model computation. Prepared as a JSON template with the help of the Builder design pattern.

Statistics calculation The prepared model was intended to be running several times in order to measure the consistency and stability of the results. To properly measure and visualize the model scores, a dedicated module for processing the results was prepared. The two main core parts can be distinguished — the plotting utilities and the statistics metrics calculation submodule.

The submodule for calculating metrics allows for producing the mean value of accuracy, model loss, percentage of FAR and FRR, number of true negatives, true positives, false negatives and false positives. The plotting submodule allows for producing the following charts:

— generic linear plot creation used to display accuracy and loss charts,

— creating the accuracy percentile histogram.

The above metrics and functionalities are designed to produce the averaged model results for multiple executions. The developed model is stable and the error is irrelevant between each execution.

3.10. Machine learning model

The machine learning model prepared and trained on the collected dataset was the part of the data evaluation final part. Google Cloud Services are able to handle the deployment of the whole system efficiently. However, this form of hosting and maintaining the infrastructure of the application comes with its price. The whole infrastructure was running for about two months and during this time generated the cost of a total $300. This amount of cash is the highest financial outlay that could have been incurred by the authors of the work since no other scholarship than PLGrid computation cluster was granted for the purpose of preparing the thesis. Longer exposure on the Web would cost extra money, that could not be afforded. The other matter is that the engineering thesis defending has its deadline specified, thus the project has been carefully thought-out in the manner of time since the beginning of the implementation. In order to meet the adopted milestones and goals, the data collection period had to follow strict deadlines. The duration of the period when the data was collected could not be extended to broader terms, which resulted in relatively small dataset.

The first remedy for a small amount of collected data was to use the transfer learning technique, described in Section 2.3. The base feature vector for transferring to the bot recognition domain was taken from the TensorFlow Hub[32] website. The model was pre-trained on ImageNet dataset[15] using the architecture of Inception V3, which is showing a great potential in the terms of the computer vision with improved performance over the previous versions, with a relatively modest computation cost [12]. The model was built with the following dimensions of 299×299×3 for the data input. The topmost layers were added for the transferring and tuning to the bot detection domain.

The first attempts were not very exciting because of the limited amount of samples. The model behaved unilaterally which means that the dominance of user's samples outshone the bot samples. The different approaches were taken to improve the performance such as linear interpolation by connecting the points in recorded sequences. Each used sequence originally consisted of many single discrete points without any additional pieces of information. Interpolation provided an order between discrete coordinates and allowed feeding the neural network with additional information.

Another considered approach was a manipulation of the input data size. The user's sequences were limited to the number of total bot sequences. This solution was aimed to balance the dataset at the cost of fewer data. The results of this approach turned out to be insufficient — because of the total amount of bot sequences, the total size of the dataset drastically shrank, which resulted in a performance deterioration. On the other hand, the duplication of the bot sequences was used. The idea was similar to the one discussed before, but instead of reducing, the number of bot samples was increased by using a single sample several times. It resulted in an artificially balanced dataset. However, this approach did not increase performance at all.

Manipulation of the distribution of labels between training and testing dataset was also considered. It was done by performing either an equalization of the total number of both types in the testing dataset and the distribution of samples between both datasets. The first solution did not affect model performance, but the second one slightly improved overall performance if the ratio was close to 50:50. When the number of training samples was significantly greater than testing ones, the accuracy decreased due to a very small number of bot samples in testing dataset.

Yet another attempt to reduce the impact of the small dataset was changing the dataset itself. The developed serializing tool made it possible to create a few datasets from recorded data with different minimal sequence length limits. Using longer sequences meant that the overall number of them would be smaller. The authors tested several ones and found out that the best performance was for a length equal to 50.

To overcome the described issue, another technique called data augmentation was used. As stated in the "The Effectiveness of Data Augmentation in Image Classification using Deep Learning" by Perez et al. article — "It is common knowledge that the more data an ML algorithm has access to, the more effective it can be. Even when the data is of lower quality, algorithms can actually perform better, as long as useful data can be extracted by the model from the original data set" [10]. The main idea of this approach is to increase the samples of the dataset by using the same sample several times, but each time slightly transforming it. The transformation is based on operations such as rotations, zooming, and flipping the original sample with some probability to do so, so each artificially generated sample should be slightly different than the others. This approach allows significantly increasing the size of the dataset. In the presented solution, it was done by development of augmentation script using Augmentor[3] package for Python. Each of the datasets (users and bot) was extended to 30,000 samples which results in a total of 60,000 samples from just 639 original samples. The ratio between the training and testing part was set to 80:20, so the network used for the training 48,000 samples and for the testing 12,000. For the optimization of the model, the optimizer called Adam was used along with the learning rate exponential decay set initially to 10^{-4}. The decay rate was equal to 0.96 and the decay steps were set to 10^5. The loss function was set to the binary cross entropy loss due to the binary classification problem.

To save time and optimize resources usage, the first approach for running the model was to execute it in a multi-threaded way. For this purpose, the module which was facilitating that was created. Soon this approach was abandoned due to the encountered problems. The main problem was that the SLURM manager is not able to split the resources in a way to the multithreaded approach to be sensible. The resources were assigned to the whole computing node on which only 2 GPU cards were able to be allocated. This caused the problem of many machine learning models trying to run on a single card.

When run on the Prometheus cluster, the better approach was to split the computation between two GPUs and run the model sequentially. For this purpose, the Mirrored Strategy distribution technique from the TensorFlow library was used whereas training is split across multiple replicas. Thanks to the split and the computing power of GPU, single execution is faster, thus can be multiplied several times.

4. Final results

The system was evaluated under the different approaches, various sets of parameters, and common techniques that help to improve the quality of the dataset and the performance of the model. Different results were compared and then selected those which have a place in this work. Below the authors present some of the approaches and compare the results. In the next sections, the concerns for the limitations of this work are raised. In the end, the elaborate conclusions and proposal for further work are described.

4.1. Results

The presented model was trained and tested using various parameters. The changes were affected by the input data as well as the neural network parameters. The final dataset properties used to train the presented model are described in Tab. 4.1. The input picture size was forced by the used transfer learning model. This limitation had an effect on resizing original pictures of sequences to the required one by the used neural network.

Tab. 4.1. Dataset properties

Human users	45
Bot users	1
Original sequences	639
Augmented sequences	60,000
Minimal sequence length	50
Model input picture size	299 x 299 [px]

Tab. 4.2. Confusion matrix values

False negatives	908
False positives	215
True negatives	5,794
True positives	5,083
False rejection rate	15.6%
False acceptance rate	3.58%

The total amount of sequences depends on the minimal sequence length, because sequences that were shorter than 50 were simply rejected. The sequence split point was established by the time interval between two consecutive actions and in the presented work was fixed to 1 second. The minimal sequence length was fixed based on several attempts. Shorter sequences than the determined ones resulted in apparently lower accuracy, when longer ones did not improve performance at all. The length of 50 seemed to be the golden mean between accuracy and amount of sequences.

The charts presented in Fig. 4.1 and Fig. 4.2 show that the model accuracy reaches the level of 90% with 0.55 value of the loss function. It can be seen that the model accuracy gap between the testing part and the training part tends to be increasing in the later epochs, which is probably caused by the amount of input data. The impact of the small dataset is also seen on the loss function chart (Fig. 4.2), where values in the training phase tend to move away from the testing phase ones.

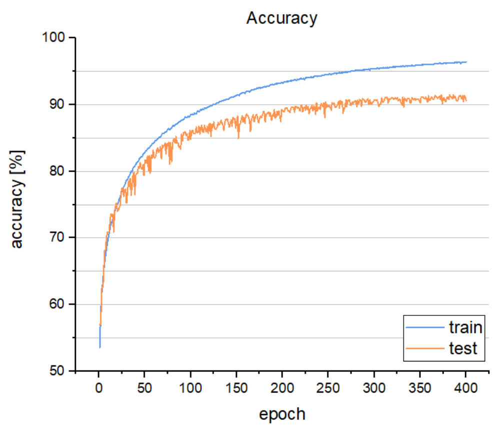


Fig. 4.1. Accuracy of presented model

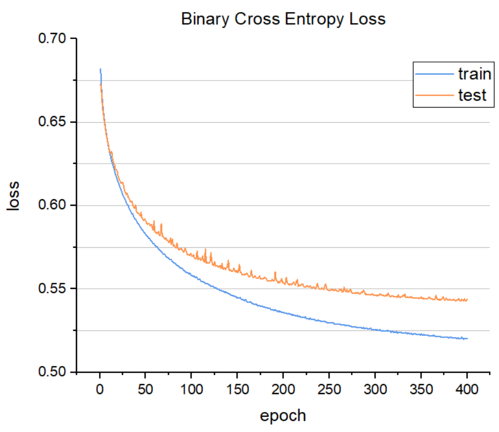


Fig. 4.2. Value of loss function used in model

The results presented in Tab. 4.2 show that the model provides correct prediction, most of the samples are classified to an appropriate category. The value of FAR is relatively acceptable; however, the FRR is significantly higher than in commercial applications.

Chong et al. [4] present a similar approach to the related problem and in many cases, they achieved very low values of FRR and FAR. In this work, using CNN's as a core of a solution, a little bit poorer results were achieved. The authors of this study consider the small dataset as the main foundation of this behavior and despite the data augmentation and other applied techniques, the results did not reach the values from the cited work.

4.2. Limitations, conditions and problems

Although the model was considered using different sets of parameters and various approaches, it did not perform as great as it was expected. Due to certain limitations, some aspects of the model performance could not have been improved. Some problems that have a great impact on the overall results are described in the following sections.

4.2.1. Dataset limitation

The main reason for this performance has to be a small dataset used in the study. The search for proper and publicly available data resulted in failure. In other related works, like Chong et al. [4] or Antal et al. [1] points at the Balabit Mouse Challenge Dataset as the most popular and comprehensive, yet still quite small data source for mouse sequences and behaviors. However, this dataset is not appropriate for the problem considered in this study.

The Balabit dataset consists of user sessions recorded on the remote machine. The data includes the timing and position of the mouse cursor of ten different users. This data does not relate to the problem given in the scope of this work — the model that can distinguish legitimate human user and non-human bot behavior is taken into consideration. This particular case is derived from the general problem of distinguishing two or more users and represents a more specific case of using mouse behavioral biometrics.

4.2.2. Custom dataset research participation limitation

Since no public dataset is available, the goal of this work was to collect exclusive and dedicated data for purpose of the study. The custom environment[9] created as a playground for research participants was designed to collect and record the mouse data, but it did not serve the responsiveness of a real commercial website, and therefore it may be causing some confusion among the subject users. By that means, data collecting was in some kind suggestive and task-oriented. Given factors could have a negative impact on the quality of the collected data.

The other issue is that participation in the study was completely voluntary and community-based. The advertisement for the ongoing study was posted on a couple of Facebook groups, which was the most available and large user community base. However, such an approach resulted in non-supervised data gathering, and therefore some user actions could not be assessed as properly executed and caused disturbances and noises in the set. The research gathered 63 unique users. Overall user mouse actions collected are equal to 334,184 rows. The number of bot actions registered and collected is equal to only 24,791 rows. Such an uneven ratio of the data made the original dataset imbalanced, which at the beginning resulted in the model making assumptions about every sequence biased towards the class of human users and after data augmentation, the results were significantly better, but not enough to use the solution on the commercial market.

4.2.3. Finance limitation

Another problem encountered when preparing the thesis was the limit of the finance intended. Because the research was planned to reach many different users it had to be deployed and hosted on trusted and reliable resources.

Cloud services are really convenient way to handle such a project — the hosting, computation, and storage resources can be acquired on-demand, with no time and commitment. In the variety of different cloud solutions, in this case, the Google Cloud Platform was selected and used. However, the cost of this kind of resources is high enough to be a limitation for the work.

4.2.4. Time limitation

Due to the time limitation, the duration of the period when the data was collected could not be extended to broader terms. Thus, the research and the voluntary participation in data collection were canceled during the further implementation of the project — the Bot Detection part[5], where the machine learning model was in build.

5. Conclusions and further study

5.1. Summary

The presented thesis focuses mainly on the data gathering system for the behavioral biometrics analysis related to the mouse actions. The introduced solution turns out successful and fulfills the main assumption of this work. Along with the data collector, the machine learning model is proposed as a potential verification step for the gathering system. To supplement the dataset with a human impersonating bot class, the custom bot is also proposed. The machine learning model is complemented by a set of tools that facilitate the work with computational cluster and also allows for seamless integration between the data collector and script with the model.

The presented results verify the data gathering system as a potential source of data for mouse dynamics biometrics researches. The system is created in a way that it can be easily deployed and tuned to the exact domain requirements. By using the Kubernetes, it is also scalable and can be even extended to the usage in related problems. The variety of collected mouse events enables the potential of the system application as a data source for recursive neural networks and also makes it possible to include other events like clicks or scroll actions into the input dataset.

The developed human impersonating bot was created mainly for verification purposes. The idea of implementation is very simple, but it can be an inspiration for more sophisticated implementations. The bot as is, can be also used in different problems — not related to behavioral biometrics. The possibility of recording the sequences allows for fast and easy configuration which can be important for researches that not focus on the bot implementation but require the bot itself.

The created tools like a serializer/deserializer allow for easily manipulating of the recorded data and transferring it into the input dataset. The tools related to the Prometheus computing cluster significantly facilitated the work with such an infrastructure and can be considered as a motivation for automation of computing and results obtaining in such an environment.

During the work on the presented solution, the steps to limit the impact of the original imbalanced dataset were taken. Many of them resulted in slightly improved performance or did not improve it at all. The key factor was the data augmentation, which resulted in a significant increase in the number of samples and also in a better performance. This technique allowed converting the unilaterally behaving model into the one whose results are promising and encouraging for further research. The linear interpolation, proposed at the beginning, was also very helpful for the model because the additional data was provided to the input which resulted in more close data domain to the ImageNet dataset. The final results suggest that the augmented dataset admittedly improves and makes the solution promising, but the tendency of moving the training curve away from the testing one should be reviewed. The authors find the small number of samples as a potential cause of these disturbances.

5.2. Further study

According to the described issues with the machine learning model, the authors find further study mainly in the improvement of the dataset. Some efforts may be taken to extend the size of the recorded data. Firstly, recording sequences of a bigger group of users may be proposed as a solution, especially to enlarge the bot users to prevent imbalance. Such a solution should improve the overall performance of the model or at least suggest other problems related to the quality of the dataset.

If the quality of the recorded samples should be improved, the collection module[9] ought to be reviewed. The main object of interest should be the method of gathering the samples. Delays and synchronization that can disturb the reliability of the data may be considered as a major area of study.

The different approach that may be taken into consideration is another machine learning model. In the presented solution, the focus was on the two-dimensional convolutional neural network, taking an example from related works like Chong et al. [4] and Wei et al. [14]. The work [4] also shows other solutions, especially recursive neural networks. Those kinds of neural networks are very popular in problems where the order and time intervals between samples have natural interpretations. The problem which is described in this work also has similar properties.

These two described areas of study are found by the authors as the major potential fields to improve performance and reliability. To deliver a safe and reliable solution to the commercial market further study is necessary.

Appendix: Links

1. Amazon Web Services — *https://aws.amazon.com/*
2. Apache Avro — *https://avro.apache.org/*
3. Augmentor — *https://augmentor.readthedocs.io/en/master/*
4. Balabit dataset — *https://github.com/balabit/Mouse-Dynamics-Challenge*
5. Bot Detection submodule — *https://github.com/Mouse-BB-Team/Bot-Detection*
6. Colorlib — *https://colorlib.com/*
7. Creative Commons Attribution 3.0 License — *https://creativecommons.org/licenses/by/3.0/*
8. Custom Bot submodule — *https://github.com/Mouse-BB-Team/Custom-Bot*
9. Data Collection submodule — *https://github.com/Mouse-BB-Team/Data-Collection*
10. Docker — *https://www.docker.com/*
11. Express framework — *https://expressjs.com/*
12. Go Lang — *https://golang.org/*
13. Google Cloud Platform — *https://cloud.google.com/*
14. Heroku Cloud Application Platform — *https://www.heroku.com/*
15. ImageNet dataset — *http://www.image-net.org/*
16. Imgur platform — *https://imgur.com/*
17. InceptionV3 transfer learning model — *https://tfhub.dev/google/imagenet/inception_v3/feature_vector/4*
18. JSON Web Token — *https://jwt.io/*
19. Java Persistence API — *https://www.oracle.com/java/technologies/persistence-jsp.html*
20. Kubernetes — *https://kubernetes.io/*
21. Node.js — *https://nodejs.org/*
22. OAuth2 — *https://oauth.net/2/*
23. PLGrid — *http://www.plgrid.pl/*
24. Pandas — *https://pandas.pydata.org/*
25. Protocol Buffers — *https://developers.google.com/protocol-buffers*
26. PyClick — *https://github.com/patrikoss/pyclick*
27. RFC 7519 — *https://tools.ietf.org/html/rfc7519*
28. SLURM — *https://slurm.schedmd.com/documentation.html*

29. Slack — *https://slack.com/*
30. Spring — *https://spring.io/*
31. Spring Boot — *https://spring.io/projects/spring-boot*
32. Tensorflow Hub — *https://www.tensorflow.org/hub*

Glossary

API : Application Programming Interface. 9, 10, 18–22, 25, 29

AWS : Amazon Web Services — cloud platform developed and hosted by Amazon. 23

CNN : Convolutional Neural Network — architecture of neural networks based on convolutions. 9, 14, 34

CSV : Comma-Separated Values. 29, 30

DNS : Domain Name System. 24

FAR : False Acceptance Rate — percentage of false recognition of users with granted access. 8, 14, 15, 30, 34

FN : False Negative — one of measures derived from confusion matrix, means a number of wrongly assigned samples to the negative category. 14

FP : False Positive — one of measures derived from confusion matrix, means a number of wrongly assigned samples to the positive category. 14

FRR : False Rejection Rate — percentage of false recognition of users that access was rejected. 8, 14, 15, 30, 34

GCP : Google Cloud Platform — cloud platform developed and hosted by Google. 9, 23, 24

GPU : Graphics Processing Unit. 32

HCI : Human-Computer Interaction. 12

JPA : Java Persistence API — official standard of ORM in Java programming language. 18

JSON : JavaScript Object Notation — notation of objects in Java Script programming language. 19, 20, 27, 30

JWT : JSON Web Token — encoded Web token that uses JSON as structural representation, described in RFC 7519[27]. 19–22

ML : Machine Learning. 13, 32

ORM : Object–Relational Mapping — idea of mapping objects from object oriented programming languages to relational structure used in relational databases. 18

ReLU : Rectified Linear Unit — activation function with the following formula: $f(x) = max(0, x)$. 14

REST : Representational State Transfer — style of Web architecture that describes stateless communication between Web services. 19

RNN : Recurrent Neural Network — architecture of neural networks based on recursion. 14

SQL : Structured Query Language — query language used mostly in relational databases. 26

SSL : Secure Socket Layer — Web protocol that ensures confidentiality and integrity of the sent data. 24

TLS : Transport Layer Security — extension of SSL that is used as a confidentiality and integrity Web protocol. 21, 24

TN : True Negatives — one of measures derived from confusion matrix, means a number of correctly assigned samples to the negative category. 14

TP : True Positive — one of measures derived from confusion matrix, means a number of correctly assigned samples to the positive category. 14

URL : Uniform Resource Locator. 18, 29

YAML : markup language, which is used in configuration files e.g. in Kubernetes. 25

Bibliography

[1] Margit Antal and Elöd Egyed-Zsigmond. „Intrusion detection using mouse dynamics“. In: *IET Biometrics* 8.5 (2019), pp. 285–294. DOI: *10.1049/iet-bmt.2018.5126*.

[2] Ahmed Awad and Issa Traore. „A New Biometric Technology Based on Mouse Dynamics“. In: *IEEE Transactions on Dependable and Secure Computing* 4.3 (2007), pp. 165–179. DOI: *10.1109/TDSC.2007.70207*.

[3] Ahmed Awad and Issa Traore. „Mouse Dynamics Biometric Technology“. In: *Behavioral Biometrics for Human Identification: Intelligent Applications*. Ed. by Liang Wang and Xin Geng. IGI Global, 2009. Chap. 10, pp. 207–223.

[4] Penny Chong, Yuval Elovici, and Alexander Binder. „User authentication based on mouse dynamics using deep neural networks: A comprehensive study“. In: *IEEE Transactions on Information Forensics and Security* 15 (2020), pp. 1086–1101. DOI: *10.1109/TIFS.2019.2930429*.

[5] Rida T. Farouki. „The Bernstein polynomial basis: A centennial retrospective“. In: *Computer Aided Geometric Design* 29.6 (2012), pp. 379–419.

[6] Wang Kaixin et al. „A User Authentication and Identification Model Based on Mouse Dynamics“. In: *Proceedings of the 6th International Conference on Information Engineering*. ICIE '17. Dalian Liaoning, China: Association for Computing Machinery, 2017. ISBN: 9781450352109. DOI: *10.1145/3078564.3078581*.

[7] Pawel Kasprowski and Katarzyna Harezlak. „Fusion of eye movement and mouse dynamics for reliable behavioral biometrics“. In: *Pattern Analysis and Applications* 21.1 (2018), pp. 91–103.

[8] Arvind Krishna. *IBM CEO's Letter to Congress on Racial Justice Reform*. 2020.

[9] Keiron O'Shea and Ryan Nash. „An introduction to convolutional neural networks“. In: *arXiv preprint arXiv:1511.08458* (2015).

[10] Luis Perez and Jason Wang. „The effectiveness of data augmentation in image classification using deep learning“. In: *arXiv preprint arXiv:1712.04621* (2017).

[11] David F. Rogers. *An introduction to NURBS: with historical perspective*. Elsevier, 2000.

[12] Christian Szegedy et al. „Rethinking the Inception Architecture for Computer Vision“. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[13] Tony Thomas, Athira P. Vijayaraghavan, and Sabu Emmanuel. *Machine Learning Approaches in Cyber Security Analytics*. Springer International Publishing, 2020.

[14] Ang Wei, Yuxuan Zhao, and Zhongmin Cai. „A Deep Learning Approach to Web Bot Detection Using Mouse Behavioral Biometrics”. In: *Biometric Recognition*. Ed. by Zhenan Sun et al. Cham: Springer International Publishing, 2019, pp. 388–395. ISBN: 978-3-030-31456-9.

[15] Karl Weiss, Taghi M. Khoshgoftaar, and Dingding Wang. „A survey of transfer learning”. In: *Journal of Big Data* 3.9 (2016). DOI: *10.1186/s40537-016-0043-6*.

[16] Roman V. Yampolskiy. „Behavioral, Cognitive and Virtual Biometrics”. In: *Computer Analysis of Human Behavior*. Ed. by Albert Ali Salah and Theo Gevers. London: Springer International Publishing, 2011, pp. 347–385. ISBN: 978-0-85729-994-9. DOI: *10.1007/978-0-85729-994-9_13*.

[17] Igal Zelfman. *Bot traffic report 2016*. Imperva Incapsula Blog 2017.