



Detecting Web Bots via Mouse Dynamics and Communication Metadata

August See^(✉), Tatjana Wingarz, Matz Radloff, and Mathias Fischer

Universität Hamburg, Hamburg, Germany

{richard.august.see,tatjana.wingarz,mathias.fischer}@uni-hamburg.de

Abstract. The illegitimate automated usage of Internet services by web robots (bots) is an ongoing problem. While bots increase the cost of operations for service providers and can affect user satisfaction, e.g., in social media and games, the main problem is that some services should only be usable by humans, but their automated usage cannot be prevented easily. Currently, services are protected against bots using visual CAPTCHA systems, the de facto standard. However, they are often annoying for users to solve. Typically, CAPTCHAs are combined with heuristics and machine-learning approaches to reduce the number of times a human needs to solve them. These approaches use request data like IP and cookies but also biometric data like mouse movements. Such detection systems are primarily closed source, do not provide any performance evaluation, or have unrealistic assumptions, e.g., that sophisticated bots only move the mouse in straight lines. Therefore we conducted an experiment to evaluate the usefulness of detection techniques based on mouse dynamics, request metadata, and a combination of both. Our findings indicate that biometric data in the form of mouse dynamics performs better than request data for bot detection. Further, training a mouse dynamic classifier benefits from external and not only website-specific mouse dynamics. Our classifier, which differentiates between artificial and human mouse movements, achieves similar results to related work under stricter and more realistic conditions.

Keywords: web bots · mouse dynamics · captchas

1 Introduction

Programs that can automatically request endpoints increase operating costs and can frustrate users. For example, web bots made popular items such as graphics cards and new game consoles unavailable for years because they were purchasing them automatically. Commercial countermeasures, such as IDS solutions, prove ineffective against web bots because these bots operate within the constraints of the targeted e-commerce website's existing APIs or user interface. For instance, web bots may mimic human behavior by navigating to a product and clicking on the "buy now" button. The problem is that the endpoints are used as intended. A defense that scans, e.g., for malicious payload in requests is pointless here. One solution is CAPTCHAs, which give users tasks that are difficult for a computer

© IFIP International Federation for Information Processing 2024

Published by Springer Nature Switzerland AG 2024

N. Meyer and A. Grochowska-Czuryło (Eds.): SEC 2023, IFIP AICT 679, pp. 73–86, 2024.

https://doi.org/10.1007/978-3-031-56326-3_6

but easy for a human to solve. However, this introduces user friction that can cost a company customers and thus revenue [9]. Modern CAPTCHAs are used together with risk assessment methods. Depending on the risk score, a certain hard CAPTCHA or no CAPTCHA at all is presented [15]. While we see this as a step in the right direction, the problem remains that such solutions are not privacy friendly as request data and biometric features are passed on to third parties. Commercial CAPTCHA providers, understandably, do not disclose which features they use for detection and which are most beneficial to identify bots.

Existing approaches are subject to various limitations, such as being closed-source, considering only request data or mouse dynamics, or having shallow assumptions in their evaluation. For instance, some approaches assume that advanced web bots only produce mouse movements in straight lines rather than curved human-like movements. Moreover, these approaches fail to address other real-world problems, such as whether website-specific mouse data or any mouse data can be used for bot detection. We address this in our paper.

Our main contribution is evaluating the usefulness of mouse dynamics for detecting bots in realistic settings. In more detail:

- We evaluate the performance of classifiers for bot detection based on mouse dynamics and compare them to the performance of using request data. We do this on a consistent dataset that contains mouse- and request data belonging to the same user. Our evaluation includes advanced bots that mimic human mouse movements utilizing third-party software.
- We show that bot detection based on mouse dynamics can benefit from not being solely trained on website-specific mouse movements, indicating that website operators do not need to train on the mouse movements of their users exclusively but can leverage third-party datasets.
- We investigate the relationship between the number of data points and the performance of bot detection, thus allowing us to determine the amount of data required for good performance and, consequently, the speed at which a classification can take place.

The rest of this paper is structured as follows: Sect. 2 discusses web bot detection using request data and mouse dynamics. Section 3 explains how and which features we used to train classifiers from related work. Section 4 describes our evaluation, how we created our dataset, and the limitations of our work. Finally, Sect. 5 concludes the paper.

2 Related Work

We divide the related work into approaches that detect bots based on request data and ones that detect bots based on mouse data. Note that there are also approaches that recognize bots based on other biometric data [5, 6], or approaches that are based on trusted platforms [8].

Modern CAPTCHA systems like hCaptcha and reCAPTCHA [15, 16] are already using biometric data like mouse movements in addition to request data.

However, they do not disclose if the detection of bots is website specific and how well which factor performs. This is most likely due to protecting business secrets and denying bot creators information about where improvements need to be made. Google itself doesn't even specify what data they use exactly for reCAPTCHA. However, there is related work that tries to break this down [19].

2.1 Bot Detection via Request Data

A simple way for bot detection is to block IPs that are known for spamming or cyber attacks, for example, using *abuseipdb*¹. Furthermore, there are many approaches for the detection of bots based on request data [11, 12, 14–16, 20].

Iliou et al. [11] present a comparison of different machine learning algorithms and combinations of various attributes used in previous literature. Their approach does not rely on cross-website tracking or using external resources like IP databases. This makes their approach simple to reimplement and validate. Their methods were tested on a year's worth of HTTP log data from MK-Lab's public web server². The data included IP addresses, the request method, the request path, referrers, user agent strings, and timestamps. The attributes mainly include request metadata that would be suitable for a privacy-friendly bot detection system, for example, the percentage of image requests or the number of total bytes per session. The authors split the bot data in their dataset into simple and advanced bots, which are determined by whether the requests have a browser agent name and, in case they do, whether the IPs have shown malicious activity before. Their results show that different sets of attributes perform best depending on the classification algorithm used. The best machine learning methods are Random Forest and Multilayer Perceptron, although the paper concludes that using an ensemble classifier that averages over all used methods would be more stable. Additionally, simple web bots can be detected very easily, while detecting advanced bots is significantly harder, with areas under the ROC curve of 1.00 and 0.64, respectively. Especially in false positive intolerant use cases, the performance of detecting advanced bots is too poor to be used in the real world. The authors conclude that future work would need to incorporate more advanced features that bots cannot easily simulate.

2.2 Bot Detection via Mouse Dynamics

There is a lot of related work in the area of authentication using biometric data using mouse dynamics [13, 17, 18].

The work by Shen et al. [18] shows that it is possible to use mouse and trackpad actions to verify the authenticity of users. For this purpose, they group mouse events such as single-click, double-click, or drag-and-drop. This data is combined with information about the current application type, e.g., web browsing or gaming, the screen area where the mouse movement occurred, the window position, and the timestamp. The data is transformed into a feature vector

¹ <https://www.abuseipdb.com/>.

² Multimedia Knowledge and Social Media Analytics Laboratory, <https://mklab.iti.gr/>.

containing data such as the time it took a user to click a button, the speed of movement, or the acceleration. Finally, three different one-class classifiers (Nearest Neighbor, Single-Layer Neural Network, Support Vector Machine) are compared, with the Support Vector Machine method performing best with false positive and false negative rates of 0.37% and 1.12%, respectively. This was achieved only when 3000 operations and 30 min of processing time for successful authentication are considered. With a more feasible authentication time of one minute, the values for FPR and FNR increase to 44.65% and 34.78%, respectively. This drastically limits the applicability of this approach, which the authors also note.

Acien et al. [1] show the feasibility of using biometric features for bot detection. They use both function-based and GAN-based mouse trajectory synthesis methods to generate training and evaluation data. Six different classifier types (Support Vector Machine, K-Nearest Neighbor, Random Forest, Multi-Layer Perceptron, and 2 Recurrent Neural Networks with Long Short-Term Memory and Gated Recurrent Units, respectively) are compared to each other, with Random Forest performing the best. Combined, their method can distinguish between humans and bots with up to 98.7% accuracy with only one mouse trajectory as input. They conclude that, compared to state-of-the-art works, the usage of mouse data has unexploited potential in the context of bot detection. While there are other approaches for bot detection based on mouse dynamics [5, 21], they are all quite similar to each other. There are even approaches and projects that try to synthesize human mouse movements [1, 2].

2.3 Bot Detection via Request Data and Mouse Dynamics

Iliou et al. [10] present a method of using request data together with mouse data for bot detection, building on their previous work on bot detection using requests [11]. When classifying mouse dynamics, they do not build on existing work but create a new model that performs the classification using a convolutional neural network (CNN) on the raw mouse positions. As they do not have a labeled bot dataset, they create bots themselves. While the general idea of their approach has merit, they perform their evaluation with advanced bots that only move the mouse in straight lines. Figure 1 shows an example of the advanced bot behavior used for evaluation in their paper.

The mouse movements created by such a bot are not realistic enough to challenge human behavior and are very easy to classify as a bot by, e.g., looking at the straightness/curvature or angular velocity of the purported mouse movements. While our approach is similar to the one presented by Iliou et al., we utilize more advanced bots for our evaluation that do not only move in straight lines and thus mimic human-like behavior more closely. We describe our advanced bot setup in Sect. 3.

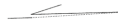
Advanced bot
✓ Heuristic hyperlink selection
✓ Advanced mouse movements


Fig. 1. Excerpt of the advanced mouse movements of [10] (Page 18, Table 7).

In summary, there is already relevant research on bot detection using biometric features, including mouse dynamics, but most of it is closed-source. Most publicly available approaches tackle the problem of bot detection independent of the website being defended, i.e., the bot detection systems are trained on a dataset that does not originate from the website being defended. Additionally, the majority of approaches consider only request data or mouse data for detection, while the ones combining both techniques only evaluate their approaches with easily detectable bots.

3 Bot Detection Using Requests and Mouse Dynamics

Our core idea is to improve the detection of web bots by using mouse dynamics in addition to request data. While request (meta)data like user-agent or screen size can easily be faked, mouse movements are continuous and contain many features, making them significantly harder to replicate. Thus an attacker would need valid, human-like mouse movements for each action. An example of several mouse movements is given in Fig. 2, which shows three different recordings of mouse movements, two from using a chat app (one activity produced by a human, the other by the advanced bot used in our evaluation) and one from using a rhythm game³ where the mouse is used heavily. This image depicts that the mouse dynamics are different per application and that advanced bots exist which do not exclusively move in straight lines.

To be consistent with prior work, we use machine learning models and request features proposed by [11] for bot detection on request data. For bot detection on mouse dynamics, we use machine learning models and mouse features proposed by [1]. Both papers are described in Sect. 2.

3.1 Request Data

Iliou et al. [11] ranked the best-performing metrics for simple and advanced bots per classification algorithm. However, some attributes used in their analysis are not suitable for this paper. For example, the authors include a Boolean indicating whether a request has a known search engine in their "Referer" header. Because we asked participants to visit the websites directly, this attribute is omitted. We use the following selection of metrics from Iliou et al.'s work [11] for our analysis:

1. The percentage of HTTP requests that led to an HTTP 4xx code response.
2. The percentage of HTTP requests that requested a CSS file.
3. The percentage of HTTP requests that requested a JavaScript file.
4. The percentage of HTTP-requested URLs that contain the previously requested URL as a subpart.
5. The total time between the first and the last HTTP request of the session.
6. Standard deviation of requested pages' depth (number of "/" in URL path).
7. Mean and Standard Deviation of times between successive requests.

³ <https://osu.ppy.sh/home>.

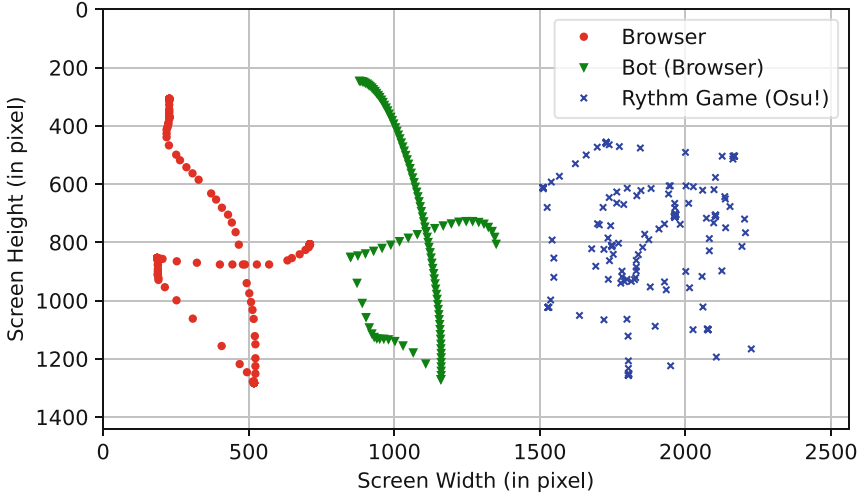


Fig. 2. Mouse dynamics example. Browser activity in a chat app on the left. In the middle movements of ghost-cursor (see footnote 7) creating a path between four given points. On the right a human playing a rythm game (Osu!).

3.2 Mouse Features

Mouse dynamic features consist of the relative x - and y -coordinates as well as a time value for each mouse event (e.g., left-click). Single mouse data points are grouped based on the following rules: They either end with a click, have a maximum of 50 data points, or span a maximum of two seconds. The features are calculated for each group. As a basis for all derived values, the time, x and y coordinates are linearly interpolated such that vectors with uniformly spaced values x'_t and y'_t every $20ms$ are generated. All indices start at zero.

In the following, we describe the additional used features engineered similarly to Gamboa et al. [7] and [4]. We include the path length from the origin, angle of the path tangent, horizontal, vertical, and overall velocity, acceleration, jerk, and angular velocity. Additionally, the type of action, length of the movement, and time needed to complete the action will be used.

The path length from the origin s'_t , i.e., the accumulated sum of previous segment lengths:

$$s'_t = \sum_{k=0}^{t-1} \sqrt{(x'_{k+1} - x'_k)^2 + (y'_{k+1} - y'_k)^2}$$

The angle of the path tangent with the x-axis θ_t is the arctangent ($atan2$ is used, which returns only values $-\pi < \theta < \pi$) of the segment at time $t > 0$. At $t = 0$ an angle of 0 is assumed.

$$\theta_t = atan2((y'_{t+1} - y'_t), (x'_{t+1} - x'_t))$$

The temporal features horizontal (v_x), vertical (v_y), tangential (v) and angular velocity (ω) as well as tangential acceleration (\dot{v}) and jerk (\ddot{v}) are computed as follows:

$$v_x = \frac{\delta x}{\delta t}; \quad v_y = \frac{\delta y}{\delta t}; \quad v = \sqrt{v_x^2 + v_y^2};$$

$$\omega = \frac{\delta \theta}{\delta t}; \quad \dot{v} = \frac{\delta v}{\delta t}; \quad \ddot{v} = \frac{\delta \dot{v}}{\delta t}$$

For each of the 9 vectors ($x'_t, y'_t, s'_t, v_x, v_y, v, \omega, \dot{v}, \ddot{v}$) the mean, standard deviation, minimum, maximum and value range (max-min) is calculated and yields the first 45 feature values.

Additionally, the time t_{total} and length s_{n-1} of the stroke (i.e. group of n data points), its straightness and jitter are computed. The time is the difference between the first and last data points' timestamps and the length can be the accumulated sum of segment lengths but using the raw instead of the interpolated data.

$$t_{total} = t_{n-1} - t_0$$

$$s_{n-1} = \sum_{k=0}^{n-1} \sqrt{(x'_{k+1} - x'_k)^2 + (y'_{k+1} - y'_k)^2}$$

Analogous to Gamboa et.al.'s definition [7], the *straightness* is defined as the ratio of the Euclidian distance between the first and last points of each group, and the total distance:

$$straightness = \frac{\sqrt{(x_0 - x_{n-1})^2 + (y_0 - y_{n-1})^2}}{s_{n-1}}$$

The *jitter* is the ratio between the original and smoothed path lengths:

$$jitter = \frac{s'_{n'-1}}{s_{n-1}}$$

In total, these 50 values make up the input vector that is computed for each mouse action group. We base our detection on whether a mouse movement is human on the features described above.

4 Evaluation

In this section, we summarize the evaluation results of our approach. We employed the best classifiers that were identified in related work. We utilized a random forest classifier [11] to analyze the request data, and for the mouse data, we also employed a random forest classifier [1]. One of the advantages of using a random forest classifier for the mouse data is its explainability, which helps in understanding how the model arrived at its predictions. We used the dataset described in Sect. 4.1 for our training and evaluation. The features used for the classifiers are described in Sect. 3.1 and Sect. 3.2. Further, we answer the following research questions:

- RQ1:** What is the performance of the detection depending on the available data, i.e., the number of requests and mouse movements?
- RQ2:** How does the performance of the machine learning model change when trained additionally with mouse dynamics from an external dataset, i.e., unrelated to mouse dynamics on our websites?

4.1 Data Collection and Augmentation

To train and evaluate bot detection approaches, we need a dataset of request data together with related mouse dynamics, i.e., the combination of requests and matching mouse movements of a user. However, such a dataset does not exist to our knowledge [10]. While some datasets on mouse dynamics exist [3], they are obtained by users that repeatedly perform specific mouse-intensive tasks. Since such a type of mouse dynamics differs from the mouse dynamics of users visiting a website, it could affect a classifier’s performance. We use this dataset in a second step to explore whether this is true.

Since no suitable dataset combining both features is publicly available [10], we need to build our dataset. For this, we invited users to visit and browse our two websites that log each request and every mouse movement. We announced our experiment with a link to our websites via a mailing list and had 322 participants visiting the first and 163 participants visiting the second website mentioned in the mail. We excluded mobile users, as well as users with no recorded mouse movements. Figure 3 shows the distribution of all users on both websites and their generated data points.

To integrate bot data we had to write our own. For this we use puppeteer. The behavior looks like this:

1. Accepting the initial prompt dialogue to start the experiment
2. Visiting the top-level pages {About, Blog, Contact/Imprint, Login, Register}
3. Visiting 10 randomly selected single blog pages
4. Visiting 100 randomly selected pages
5. Registering an account

All actions are configured to wait for the target element to be visible and clickable, scrolling it into view, if not. A random delay between 0 and 2 seconds is applied before each action. The behavior should reflect a scraper that does not scrape at full speed and in a specific order, e.g., width search.

Further, we distinguish between a basic mouse bot and an advanced mouse bot. The basic mouse bot moves the mouse on a direct path and at a constant speed to the target (links, blog posts, ...). The advanced mouse bot does not do this but uses bezier curves implemented in the popular javascript library `ghost-cursor`⁴, which promises human-like mouse movements. An example of such movements created by `ghost-cursor` is depicted in Fig. 2. We sample as many bot users as human users in the experiment.

⁴ <https://github.com/Xetera/ghost-cursor>.

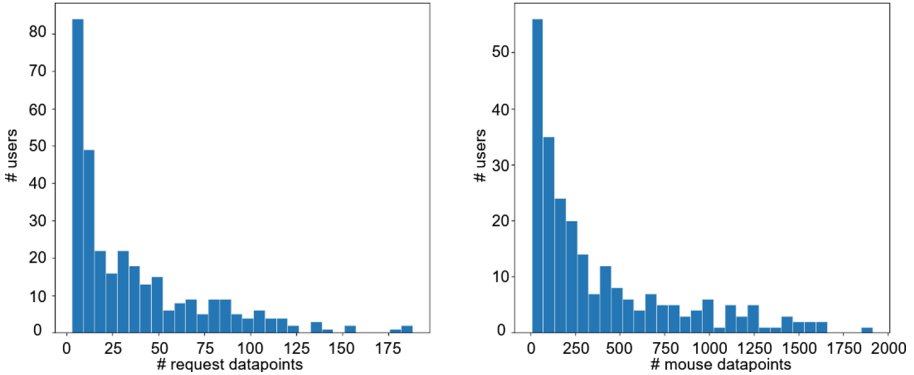


Fig. 3. Distribution of users in terms of data point count

4.2 Results

This section presents our results. Limitations are described in Sect. 4.3.

RQ1 - Bot Detection Performance. The performance of our model for the detection of bots depends on the data available. With more data available, the request data model’s evaluation metrics show increasingly good performance. Table 1 shows the detailed results. Note that there are many cases where fewer data points than the limit are available (cf. Figure 3). The mouse data model generally performs better the more data is available. Table 2 lists the performance for different amounts of data points per user. A big advantage of this approach is that more data points can potentially be acquired in a shorter amount of time compared to request data. For example, when sampling at 30 events per second, as this work’s implementation does, it only takes on average 1.66 s (50 samples) to capture the number of data points needed to surpass the request data model’s performance. When, for example, considering a potential application for a CAPTCHA, this time does not represent a significant disruption of most user interactions, e.g., filling in a registration form.

Parameter Tuning We used combinations of the following parameters to determine the best random forest parameters for bot detection empirically. Note that the number of bots and users in the dataset is the same, i.e., the same number of sessions. For mouse movements, we use the advanced bot that mimics human mouse movements.

1. Number of estimators (10, 50, 100, 150, 200, 1000)
2. Maximum number of features (None, log2, sqrt)
3. Maximum tree depth (None, 1, 2, 3, 4, 6, 7)

Tables 3 and 4 show the 10 best-performing combinations for mouse and request data. The data is sorted by accuracy. The additional scores Precision,

Table 1. Request data model performance with varying amounts of data points per user

Data Points/User	Acc	Precision	Recall	AUC	Time
200	0.980	0.980	0.980	0.982	0.209
100	0.970	0.961	0.980	0.985	0.209
No limit	0.960	0.960	0.960	0.972	0.215
50	0.950	0.941	0.960	0.976	0.208
20	0.910	0.918	0.900	0.975	0.221
5	0.900	0.885	0.920	0.945	0.220
10	0.890	0.842	0.960	0.956	0.220
4	0.880	0.913	0.840	0.922	0.211

Table 2. Mouse data model performance with varying amounts of data points per user (Advanced Mouse Bot)

Data Points/User	Acc	Precision	Recall	AUC	Time
No limit	0.966	0.964	0.968	0.993	0.449
50	0.933	0.943	0.922	0.979	0.124
200	0.930	0.951	0.906	0.979	0.189
100	0.920	0.942	0.895	0.975	0.149
20	0.855	0.846	0.868	0.949	0.114
10	0.850	0.862	0.833	0.903	0.120
5	0.846	0.909	0.769	0.916	0.099
4	0.826	0.895	0.739	0.879	0.098

Recall, AUC, and training time are computed as well. Their values of the top-performing results lie close together except for training time. The different values for the number of estimators and the maximum number of features for split consideration perform very similarly. For request data, the values for the following experiments were chosen to be 100 and *None*, respectively, as their result had the same accuracy and AUC as the top result. Analogously for the mouse data result, 200 and *sqrt* were chosen. All results have in common that no restriction to the decision trees’ maximum depth is applied, which is also the default value of *scikit-learn*’s implementation. This is expected as the tree depth is directly correlated with the ability to classify multi-dimensional input data.

Basic vs. Advanced Mouse Bot. The first direct comparison used all available human mouse data and the generated basic and advanced mouse data for training and testing datasets. Table 5 shows that the model performs better in every aspect and can classify inputs more reliably when using data generated only by the basic mouse bot with linear movements. This is expected as the bots’ linear movements are uniquely identifying properties that result in very specific out-

Table 3. Model accuracy for different parameters (request data)

Features	Estimators	Acc	Prec.	Recall	AUC	Time
None	1000	0.910	0.887	0.940	0.973	4.348
log2	200	0.910	0.887	0.940	0.973	0.732
log2	1000	0.910	0.887	0.940	0.975	3.965
None	100	0.910	0.887	0.940	0.973	0.486
sqrt	1000	0.910	0.887	0.940	0.972	3.912
log2	100	0.910	0.887	0.940	0.971	0.389
log2	150	0.910	0.887	0.940	0.972	0.694
None	50	0.900	0.870	0.940	0.971	0.273
sqrt	200	0.900	0.870	0.940	0.972	0.727
None	150	0.900	0.870	0.940	0.972	0.703

Table 4. Model accuracy for different parameters (mouse data)

Features	Estimators	Acc	Prec.	Recall	AUC	Time
sqrt	150	0.967	0.964	0.969	0.994	11.305
log2	1000	0.966	0.962	0.970	0.993	70.796
sqrt	200	0.966	0.962	0.970	0.994	5.899
sqrt	50	0.966	0.964	0.968	0.993	9.280
sqrt	100	0.966	0.962	0.969	0.994	9.815
sqrt	1000	0.966	0.960	0.971	0.994	82.889
log2	200	0.964	0.963	0.965	0.993	20.381
log2	150	0.962	0.963	0.960	0.993	5.950
None	150	0.961	0.954	0.969	0.991	91.586
None	200	0.961	0.953	0.969	0.991	110.588

comes for many input features. The model only correctly differentiated between humans and bots 96.6% of the time but still has a very high AUC.

Combining Mouse and Request Data for Advanced Bot Detection. Since we have a dataset containing request data and a user’s matching mouse dynamics, we can explore whether a bot detection system that combines both mouse and request data may improve the performance compared to using them individually. Therefore, we apply the classifiers mentioned above to the mouse and request data individually. Afterward, we combine the calculated predictions of the two classifiers and average them to determine the final result. We split the data into training and test sets on the user level, i.e., each user instance (human or bot) is only part of either the training or test set. We use two test ratios for this experiment, namely 0.1 and 0.2. Table 6 shows the overall performance, the precision of 1.0 was omitted from the table for readability. The most valuable

Table 5. Simple and Advanced Mouse Data Performance

Scenario	Acc	Prec	Recall	AUC	Time
Basic mouse	0.995	0.997	0.994	1.000	0.857
Advanced mouse	0.966	0.962	0.970	0.994	1.293

difference is that there are no false positive results, while the false negative rates of 0.272 and 0.327 are higher in contrast to using the classifiers separately. However, the lower false positive rate and same overall performance favor using the combined approach as it is a priority not to disrupt the user experience [9].

Table 6. Combined Mouse and Request Data Performance

Test ratio	Acc	Recall	AUC	TP	FP	TN	FN
0.1	0.960	0.953	0.976	122	0	22	6
0.2	0.950	0.940	0.970	252	0	49	16

RQ2 - Using Unrelated Mouse Dynamics for Training. We used Antal et al.’s dataset [3] to compare the performance to different real-world data. Inputs from 21 users were collected during their normal computer activities on one desktop and 20 laptop devices. Both mice and touchpads were used. Their raw mouse movement and interaction data are preprocessed similarly to the experiment’s data. The whole dataset yielded 1.54M input vectors. The initial assumption was that these mouse movements do not match the interaction with our website and the performance decreases when using these mouse movements. However, when trained additionally with the external dataset, the accuracy increases to 99.71%, and the values for FPR and FNR decrease to 0.55% and 0.15%, respectively. This indicates that the origin of the mouse movements is not important with regards to their effectiveness.

4.3 Limitations

The strongest limitation of our approach is that we only have access to a synthetic bot dataset, similar to approaches like [10]. Further, we create bots using fitting third-party projects. However, an external, labeled dataset with bot- and real-human traffic would be more suitable. The lack of such a realistic dataset leads to the performance of our model likely being worse outside our lab setting. Bots may act more camouflaged, e.g., by using recorded mouse movements. Those bots would probably escape detection. However, this is universal. Bots that behave completely like humans cannot be distinguished from humans. At the same time, making a bot behave like a human increases the costs for an attacker because the bot cannot work at full performance, e.g., scrape all data available or monitor a website for a long duration.

Further, while we were able to show that mouse data can easily be used for the detection of bots, another limitation is the focus on mouse data. This excludes users who interact without a mouse, e.g., only via keyboard, mobile devices, or screen readers.

5 Conclusion

We demonstrated the value of incorporating both mouse dynamics and request data when detecting bots. Unlike previous research that relied solely on one of the two data types or made unrealistic assumptions about the capability of advanced bots, we used a consistent dataset that included both mouse movements and request data belonging to the same user. Furthermore, we utilized a third-party library to create bots that performed human-like mouse movements. We used classifiers that performed best in literature for these tasks. We achieved better results with similar performance but in a more realistic setting. Thus mouse dynamics remain a useful tool for identifying even advanced bots. An interesting finding was that mouse data from third-party sources can be used to train the classifiers while achieving similar performance, thus simplifying the usage process. By leveraging third-party mouse data, operators can minimize the need to save and train on potentially sensitive user data. In the future, we intend to test our approach on a larger e-commerce dataset. Further, we want to consider more combinations of alternative approaches to bot detection, e.g., by including typing behavior as well as touch events from smartphones.

References

1. Acien, A., Morales, A., Fierrez, J., Vera-Rodriguez, R.: BeCAPTCHA-mouse: synthetic mouse trajectories and improved bot detection. [arXiv:2005.00890](https://arxiv.org/abs/2005.00890) [cs] (2021)
2. Akrouf, I., Feriani, A., Akrouf, M.: Hacking google reCAPTCHA v3 using reinforcement learning. arXiv preprint [arXiv:1903.01003](https://arxiv.org/abs/1903.01003) (2019)
3. Antal, M., Denes-Fazakas, L.: User verification based on mouse dynamics: a comparison of public data sets. In: 2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics, pp. 143–148. IEEE (2019)
4. Antal, M., Egyed-Zsigmond, E.: Intrusion detection using mouse dynamics. *IET Biomet.* **8**(5), 285–294 (2019)
5. Chu, Z., Gianvecchio, S., Wang, H.: Bot or human? A behavior-based online bot detection system. In: Samarati, P., Ray, I., Ray, I. (eds.) *From Database to Cyber Security*. LNCS, vol. 11170, pp. 432–449. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04834-1_21
6. Dee, T., Richardson, I., Tyagi, A.: Continuous transparent mobile device touch-screen soft keyboard biometric authentication. In: 2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID), pp. 539–540. IEEE (2019)
7. Gamboa, H., Fred, A.: A behavioral biometric system based on human-computer interaction. In: *Proceedings of the SPIE*, vol. 5404, pp. 381–392 (2004). <https://doi.org/10.1117/12.542625>

8. Gummadi, R., Balakrishnan, H., Maniatis, P., Ratnasamy, S.: Not-a-bot: improving service availability in the face of botnet attacks. In: NSDI, pp. 307–320 (2009)
9. Heath, N.: Expedia on how one extra data field can cost \$12m (2010). <https://www.zdnet.com/article/expedia-on-how-one-extra-data-field-can-cost-12m/>. Accessed 18 Oct 2021
10. Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S., Kompatsiaris, I.: Detection of advanced web bots by combining web logs with mouse behavioural biometrics. *Digit. Threats: Res. Pract.* **2**(3), 1–26 (2021)
11. Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S., Kompatsiaris, Y.: Towards a framework for detecting advanced web bots. In: Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019. Association for Computing Machinery, New York (2019)
12. Jonker, H., Krumnow, B., Vlot, G.: Fingerprint surface-based detection of web bot detectors. In: Sako, K., Schneider, S., Ryan, P.Y.A. (eds.) ESORICS 2019. LNCS, vol. 11736, pp. 586–605. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-29962-0_28
13. Jorgensen, Z., Yu, T.: On mouse dynamics as a behavioral biometric for authentication. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 476–482 (2011)
14. Li, X., Azad, B.A., Rahmati, A., Nikiforakis, N.: Good bot, bad bot: characterizing automated browsing activity. In: 2021 IEEE symposium on security and privacy (SP), pp. 1589–1605. IEEE (2021)
15. Liu, W.: Introducing reCAPTCHA v3: the new way to stop bots (2018). <https://developers.google.com/search/blog/2018/10/introducing-recaptcha-v3-new-way-to>. Accessed 20 May 2021
16. Machines, I.: Stop more bots. start protecting user privacy (2018). <https://www.hcaptcha.com/>. Accessed 20 May 2021
17. Sayed, B., Traoré, I., Woungang, I., Obaidat, M.S.: Biometric authentication using mouse gesture dynamics. *IEEE Syst. J.* **7**(2), 262–274 (2013)
18. Shen, C., Cai, Z., Guan, X.: Continuous authentication for mouse dynamics: a pattern-growth approach. In: IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012), pp. 1–12 (2012). <https://doi.org/10.1109/DSN.2012.6263955>
19. Sivakorn, S., Polakis, J., Keromytis, A.D.: I’m not a human: breaking the google recaptcha. *Black Hat* **14** (2016)
20. Suchacka, G., Cabri, A., Rovetta, S., Masulli, F.: Efficient on-the-fly web bot detection. *Knowl.-Based Syst.* **223**, 107074 (2021)
21. Wei, A., Zhao, Y., Cai, Z.: A deep learning approach to web bot detection using mouse behavioral biometrics. In: Sun, Z., He, R., Feng, J., Shan, S., Guo, Z. (eds.) CCBR 2019. LNCS, vol. 11818, pp. 388–395. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-31456-9_43