Web Bot Detection Using Keyboard Behavioural Analysis

Vasvi Sharma, Dr. Siddhartha Chauhan*

National Institute of Technology Hamirpur, Hamirpur-Tauni Devi road, Hamirpur, 177005, Himachal Pradesh, India

* Corresponding author

doi: https://doi.org/10.21467/proceedings.178.10

ABSTRACT

Ever increasing technology and internet availability resulted in a huge rise in web bots. Users utilize web bots with good or malicious intent. The increase in web bot traffic has raised concerns for the safety and security of the web. To address this issue, there was a rise in the development and implementation of bot detection mechanisms. To ensure the safety and security of the web, various methodologies have been applied, including algorithms based on temporal and behavioral characteristics, CAPTCHAs, etc. With advancements in bots mimicking human fingerprints, mouse movements, and feeding CAPTCHAs, it became very easy to evade these detections. In this paper, an additional layer of security is introduced that utilizes keyboard behavior analysis to detect bots. The proposed algorithm works on collecting and storing data related to each keystroke, which includes records based on timestamps, key names, key hold time, and key time difference. The algorithm processes the recorded data through various conditions and parameters to conclude the detection. This algorithm works on top of the other detection mechanisms, like weblog and mouse movement detection. The proposed algorithm is implemented on a publicly provided data set to measure its effectiveness and accuracy. The findings prove that the algorithm works as an effective layer for detecting bots through the input mechanism.

Keywords: Web bot, Keystroke Dynamics, Key Hold Time.

1 Introduction

Increasing easy access and availability of the internet results in an enormous growth in usage of the web. A person's daily activities often involve using websites, whether for social media, entertainment, online banking, online education, or other purposes. With increased web usage, there was an increasing need for web bots. A Web bot is also known as a web agent or intelligent agent. It is a software program or tool that follows a specific algorithm and carries out (usually autonomously) specific tasks on the web [1]. Web bots have proven to be very useful in completing a given task at a faster pace than humans. They increase the efficiency, and performance of a task while reducing the labor cost for an individual or organization. Additionally, unlike humans, the all time availability of a bot makes it easily customizable to an organization's needs and helps them to give a better user experience.

Web bots are capable of being utilized and executed in numerous domains like business, medicine, sports, and other related fields. They can be used for automating repetitive tasks, web indexing, browsing, extracting data from any web page, validating web pages for working of all the hyperlinks, blacklisting or blocking various emails, comparing products on numerous web pages, and web search engine crawlers. Hence, web bots prove to be an integral part of the web without which various tasks are difficult to process.

As a result, there was a rise in bots with malicious intent called Malicious bots. Malicious bots are pre programmed such that they can implicitly find and exploit the vulnerable areas present in a website. For example, websites that are outdated and are not updated to increase the security [2]. These bots can be used for various other purposes such as spam which can be later used to create phishing websites or multiple accounts, and email addresses to be sold. They are commonly used to extract email addresses and other

Proceedings DOI: 10.21467/proceedings.178; Series: AIJR Proceedings; ISSN: 2582-3922; ISBN: 978-81-984081-8-1

content from websites, as well as to increase engagement and followers on various social media platforms. [3]. They are used to access the data by scraping the credential information from another web page. Recently, there have been a lot of attacks that were found to be following these methodologies. For example, ride sharing companies scrape pricing and vehicle information from competitors' websites [4, 5]. Bots are used to buy bulk tickets for an event and sell them at a higher price, eliminating a fair chance for the public to buy tickets. boosting the price of tickets due to a spike in the ratio between web page visitors to web page visitors booking from it. Bots are used to create traffic and congestion, which in turn affects the user's experience on the page. They are also being used for Distributed Denial of Service (DDoS) attacks, data scraping, data scalping, and for conducting fraud. Malicious bots pose a threat to a lot of businesses. There was a question of safety and privacy for all that is present on the web. With the daily increase of new users, businesses, and web pages joining the web, it became a huge risk.

Web bots take up a significant amount of traffic which is more than 37.9% of the internet [6]. Out of the total traffic malicious bots take over a huge percentage [5]. This poses a serious issue. To mitigate this risk, an investment was made in the bot detection mechanisms. Bot detection works on analyzing traffic to the website and identifying bots that are present in it. Various methodologies have been introduced to detect bot like detection based on mouse movement, web logs, abnormal duration of sessions, and high view rate of the page. For example, to differentiate the mouse movement of bots and humans various behavioral characteristics are used to identify. A human randomly moves a mouse while a bot moves in a straight line to reach a point. A human mouse movement accelerates or decelerates from one point to another while a bot moves with zero or constant acceleration. For a social media bot account history and engagement activities present like comments, likes, shares and saved posts are checked. Various algorithms also include machine learning or neural networks for predicting bots. For instance, algorithms can be utilized to identify a specific user's typing habits, such as the speed at which they type, the amount of pressure applied to each key, and the way they press certain combinations of keys. There are also various techniques designed to prevent bots that are CAPTCHAs, Web Application Firewalls, and Multi Factor Authentication (MFA). CAPTCHAs stands for Completely Automated Public Turing test to tell Computers and Humans Apart [7]. They automatically pop up in different places for the users to fill in. It presents a distorted text that a bot has trouble reading compared to the user who can easily read it. The user reads and writes the text in the given input box provided. If it is correct then the user can proceed further else it has to be rewritten correctly.

Development in technology yields advanced level bots that undetectably pass through all the techniques and algorithms. Bot detection becomes more challenging as bots can mimic human like mouse movements, have human like fingerprints, correctly fill in CAPTCHAs, and go through firewalls. Some bots are smart enough to modify and manipulate the elements through which bots were detected. For instance, the bot detection script looks for the word "selenium" or "web driver" in a window object or searches for a document variable called "$cdc" or "$wdc". So, with the help of a simple function, these word or document variables can be easily changed or hidden making it undetectable. If a bot is undetectable through mouse movement and web logs for example, taking another possible scenario where the bot fills up a form and the mouse movement and web logs techniques fail to detect a bot. Also in the case of a bot is only used in typing. Therefore, there is an immense requirement of input analysis of keystrokes to detect the presence of a bot. The use case applications are keystroke detection in CAPTCHAs, sign in in websites, typing in a search engine, in search explorer in websites, or any place where typing is required by the user. Adding input detection would strengthen the bot detection mechanism and would make it more challenging for a bot to slip through.

Proceedings DOI: 10.21467/proceedings.178
ISBN: 978-81-984081-8-1

Series: AIJR Proceedings
ISSN: 2582-3922

The proposed algorithm detects bots or humans through input provided by the user. Collecting the timestamp of each keystroke being pressed or released, the time the key was being pressed, the time between one key release to another key press, how fast the input was given, and how much time it took to give the entire input. Additionally, there is an inclusion of the possibility of the input being pasted, adding further scenarios of whether the entire text was printed or some portion of it was pasted. Further, a check is conducted to assess if any keyboard shortcut key was pressed and the effect it would have on the input. Developed an algorithm to collect and process this information and another script was developed which would then run on top of it to process this data further on various parameters to conclude whether the user or a bot provided the input. This algorithm was then later applied to a publicly available dataset. This data set was processed further using a set of codes to make it compatible for the algorithm to work on. Implementing the algorithm to refine the dataset to get how accurately it can detect bots.

This mechanism is particularly effective as its detection is independent of any package, library, or tools(e.g. document variable) present. Hence, it would work even if there are no traces of any package being used in the script. For example, The presence of a domain variable or selenium or web driver keyword won't affect the detection mechanism. Various detection programs are dependent on checking account history and activities, this technique does not apply to all bot activities. To detect in all these areas additional input detection would be a great be of immense help. Other than that, it would easily work if a bot can generate human like input i.e., input given by the bot but identified as received from the keyboard. Applying machine learning models in the background affects the system's performance. Hence, ample of websites would not prefer applying these models, making them risk prone. The proposed algorithm won't consume huge space and would help reach out to multiple web pages. It identifies a bot that creates an intentional time difference in any of the parameters (listed in methodologies) to escape detection. Hence, this algorithm would be significant in detecting bot activity efficiently and in a wide domain.

This method provides a new way to detect bots. This method provides high accuracy without adding load on the web pages, unlike other detection mechanisms that use machine learning algorithms, artificial intelligence, and deep learning. This algorithm does not have huge time and space requirements. This algorithm is independent of any technology or packages or methodologies used by a web bot making it quite effective, consistent with time and the latest technology. It is highly compatible with other detecting modules. It can be combined with other new approaches which can be based on various latest technologies like machine learning. It can be combined with detecting modules not only related to input detection but also with different detecting techniques like mouse movement, and weblogs. The proposed algorithm is created and developed from the basics and is a new approach.

2 LITERATURE REVIEW

Web bots aim to efficiently differentiate between humans and bots. As new malicious bot technologies emerge, there are corresponding advancements in bot detection. Bots range from simple to advanced level [6]. Where each level adds more functionality to the bot making them harder to be detected. Starting with a simple bot created using basic scripts, it is easy to detect them by comparing their fingerprints. [8]. These fingerprints include checking the header request, cookies, web sessions, etc. Development in bots could be seen after the introduction of automation tools. These automation tools could be created by using various packages like Selenium. Using selenium a user can mimic mouse movements, provide input to the web page, select elements, and carry out various functions. This helped the bot to mimic humans and interact with other web entities. But this still could be identified using their fingerprints [9] With more technology, these bots have become more complex making it difficult to identify. For example, the bots can use normal browsers similar to a human. With the different divisions of bots present, the individual portion of web

bots are observed, simple web bots account for 26.4% of traffic, bots that are more complex and can use "headless browser" functionality account for 52.5%, and sophisticated web bots which are equivalent to the advanced web bots account for 21.1% [6].

Keeping this in mind, there were various advancements in the methodologies used to tackle the bots. The proposed algorithm in this paper will discuss the development of detecting a bot by input provided by the user. One of the early methods used a way to differentiate between humans and bots using behavioral biometrics like mouse movement and weblogs. It is based on passive monitoring, which provides various benefits like not relying on a single checkpoint but rather continuously monitoring the entire session, making it less likely to be wrong. It helped as it created no additional work for the user, unlike CAPTCHAs which cause hindrance during the access process [10]. This method was based on applying continuous tests with the help of two main components, a server side classifier and a web page embedded logger. The logger is used as a JavaScript snippet running on the website which is used to record all users' actions and send them to the server detector. The foundation of the model is a machine learning based classifier, which is specifically trained on the data for binary classification, which is divided into two results known as human or bot. After getting a decision from the model the server decides the next step which is to accept or reject the form submission. [5].

While compared to the newer bot detection mechanism that can detect advanced level bots using weblogs and mouse movement through machine learning and deep learning techniques. These techniques carry out the result by measuring each characteristic separately using advanced tools and applying different techniques such as multiple machine learning models and fusing them to get the desired result. Biometric plays an important role in detecting a person. Biometric characteristics play an important role as it play an important aspect in identifying each person differently as it is unique for each individual. So, this becomes an instinctive choice to be used in detection mechanisms. There are various kinds of research carried out from keystroke dynamics to using the pressure applied by users while typing to authenticate them. These techniques started developing after it was shown that it was possible to use them for identification. The rand report in 1980 was coined as revolutionary research work done in the area of keystroke dynamics. It was inspired by the unique rhythm of each individual when it was used to send a telegraph [11]. The idea of digraph was introduced from it. Digraph refers to a pair of keystrokes hit consecutively and the time between the using the first key to the second key [11, 12]. A study was done that stored digraphs for various candidates and measured them on kurtosis and mean variance. Consequently, a test was conducted to identify different users. This study stated that it was possible to differentiate users based on their keystrokes. Following the Rand report many more experiments and studies were conducted that helped in providing the foundation of digraphs in identifying each user's typing style or signatures [11]. Later, in 1986 a user authentication method was developed through which each user needed to type their name. Later on, a patent was developed that described the importance of latencies and pressure in the measurement of keystroke behavior. Subsequently, there was research to make it more feasible. The first research was conducted in 1994 to analyze and study neural networks as a method of classifying vectors. Further, studies claimed to add keystroke time duration as a major contributing factor for keystroke authentication.

Keystroke techniques are applied either statically or dynamically. It is further divided based on its application on either fixed length texts or free length texts. It is a biometric feature that has the major advantage of not needing any additional measuring equipment like a sensor other than a keyboard [13]. For example, a static approach might prove to be a robust method for a password but not reliable for continuous security while in the dynamic approach, the user's behavior is measured continuously. Keystroke Dynamics are applied in many use cases. Keystroke dynamics is the process of examining the way an individual detects

Proceedings DOI: 10.21467/proceedings.178
ISBN: 978-81-984081-8-1

Series: AIJR Proceedings
ISSN: 2582-3922

a user's natural rhythm typing pattern by monitoring the keyboard inputs [4]. It is applied to various fields such as user authentication for passwords. It is also being used to detect bots by training the model on various datasets and applying various supervised classifiers like Random Forest, Support Vector Machine, Gaussian Naive Bayes, and including human and synthetic samples to get a learning framework. This experiment is used to demonstrate that synthetic samples are realistic [14]. With this, there is a massive improvement in bot detection. Some of the work proposes statistical approaches to generate synthetic data for the biometric keystroke using user dependent models or universe which help in training the system better for bot detection. Some of the techniques focus on user verification by applying other learning models like Manhattan Distance, Manhattan Filtered Distance, Manhattan Scaled Distance, etc. Another proposed mechanism is to work on impostor patterns. It proposes to re train the framework where novelty detector is retrained using the imposter patterns to increase authentication accuracy. Here, it retrains the imposter pattern with support vector data description and vector quantization for novelty detection [15]. Another way through which each user's profile can be developed using the pressure which is applied while pressing and releasing a key. The research carried out on Dynamic Keystroke Pressure Based showcases the ability to continuously identify and verify identity throughout the period and achieves high accuracy under controlled and strict conditions [16]. Keystroke when applied as a second layer in authentication will help a lot in higher accuracy for verification and can be used as an alternative to CAPTCHAs. It is a web based active and functioning platform present in a browser environment that enforces second layer security as mentioned [17]. Some of the other approaches use a new bio statistic feature to detect web bot activity. Using a combination of supervised and unsupervised machine learning algorithms a proposed web detection model is developed [18]. One of the latest research papers works on analyzing biometric and fairness benchmark evaluation [19]. It analyzes the various obstacles and issues that are present in the keystroke dynamics and it measures the fairness of that evaluation. For example the development of keystroke biometrics is hampered by the variability of experimental techniques and measurements as well as the small size of the databases used in the literature, which prevents direct comparisons between various systems [19]. It then proceeds to present a new framework that benchmarks KD based biometric verification. The other proposed mechanism is to add a keystroke dynamic as a part of multi factor authentication. The two step verification adds a delay in the workflow and increases friction. On the other hand, adding keystroke dynamics as one of the authentication factors would decrease the friction present and would improve the security of the authentication [20]. Another novel method was developed to authenticate users through keystroke analysis using bigram embedding [21]. This is achieved neural network based transformer that can distinguish between bigrams. In addition to this there is an application of supervised learning techniques to compute embedding for both bigrams and users [21].

3 METHODOLOGY

3.1 Terminologies Used in Algorithm

The proposed algorithm works on live data collection from the user and starts data processing on the collected data. Some of the few important features that are collected and processed in the workflow are mentioned below [22].

1. Key Press Time is the time at which a key is pressed by the user
2. Key Release Time is the time at which a key is released by the user
3. Key Hold Time is the time between release and press of the same key
4. Key Time Differences is the time between the release of the first key and the Key Press of the next key

5. Total Input Time is the Total time to fill the input

As it can be seen in figure 1 once the key named "G" is pressed it automatically records the key press time for key "G". When the key "G" is released by the user, it again records the key release time for "G". The same procedure is followed for the next key press, release for the key "H". The time between press and release of key "G" is called Key Hold Time, while the time between the key release of "G" and key press of "H" is called Key Time Difference. The total time refers to the time taken from the initial key press ("G") to the release of the last key ("H").

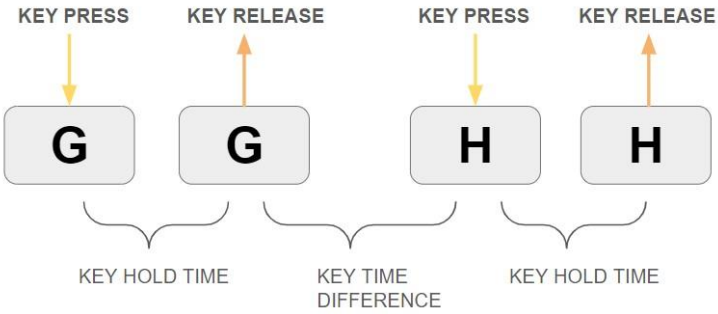

Figure 1. Keystroke Terminologies

3.2 Working of the Algorithm

The proposed algorithm is based on the assumption that the operating system is Windows and other detection based methodologies are working simultaneously, like simple mouse movement and web logs. Cases that can be easily detected by the other simultaneous methods are not taken into consideration as it would be redundant. For a better understanding of the overall workflow of the algorithm refer to figure 2. It provides an entire working mechanism of the algorithm from beginning to end inclusive of each decision making and processing. The algorithm begins to operate as soon as it detects a key press. A cell being selected or active does not affect the algorithm, timestamps are initiated to record when an input is given. It provides accurate data when the user is active.

First, the algorithm begins by focusing on continuous data collection. It starts storing all the key press timestamps and key release timestamps for each key that is pressed regardless of its reflection in the GUI input. For example, pressing a TAB key. This includes all the monographs, digraphs, or special keys that are pressed. Digraphs are used as shortcut keys like ctrl+v which refer to pasting text. To execute the methodology multi threading is required. The first thread is created to provide a user input field. This thread continuously records all user input. Concurrently a second thread is created to record all the key press time and key release time for each keystroke made by the user. This thread helps in storing data. The second thread closes with the first thread. Terminating both the threads leads algorithm to the data processing stage. The collected data is processed for further use. It calculates key hold time for each key and key time differences between each key and stores them separately. Additionally, it stores the total time taken to write the input. Total time taken includes time from when the first key was pressed to when the last key was released. Successful processing of the required data advances the algorithm to the next stage of authentication.

It begins by verifying the authenticity of the provided input. A simple script that won't be able to generate authentic keyboard input signals. It is verified through the stored key press and key release data. If the data is not present then it is concluded to be filled in by a script. This condition results in the conclusion that it

Proceedings DOI: 10.21467/proceedings.178
ISBN: 978-81-984081-8-1

Series: AIJR Proceedings
ISSN: 2582-3922

is a bot. This also removes the possibility of erroneously detecting entire text deletion by checking key presses.

In the case where a bot can generate authentic signals, it processes and analyzes the present data. It checks the total time taken to provide the input. It divides the algorithm into two branches based on the result. A different algorithm is followed if the time taken is less than or more than one second. If the time taken is less than one second it is more prone to being a bot if the input is of equal or more than two characters. It is not humanly possible to write a sevenletter word in less than a second. If the time taken to fill it is more than one second, it makes it slightly less likely to be bot. So it follows a different procedure for both cases.

Starting with case 1 where the time taken is less than one second. It begins by checking the length of the typed input. It divides case 1 into further subtypes based on the length of the input. If the length of the input is more than or equal to four letters it moves on to another set of procedures i.e it checks whether the paste command is being used or not. If the paste command is detected then it moves on to check the authentic usage of the paste command. The usage of the empty paste command helps the bot bypass detection. If nothing is pasted then it is declared to be a bot. As an average user cannot write faster than this. Characters typed in one second are taken after taking into consideration the average typing speed of a person. If the length is less than four, the system checks for digraph keys that are pressed. It ignores all the digraphs except the one used to paste text, "ctrl+v". The scenario where a user inputs the text using the paste command. An input can be filled in less than a second using paste. If the above case is false, and a user fills more than three letter input in less than a second, results it being a bot. If the paste option is used then it carries out further checks. It starts checking the data stored in Key Hold Time and Key Time Difference. It first checks if all the elements in both of them are not the same to their list. For example, in key hold time, the majority of the list contains a hold time to be 0.075 seconds i.e. constant. This case is declared to be a bot as a person cannot constantly hold or press the next key using a constant time throughout. If it is not the case it moves on to check cases. If the time for holding the key is zero or near zero (in decimals) for the majority of the keys then it is declared as a bot. Near zero values are considered due to the latency of the hardware after receiving the signal or intentional delays added in any of the parameters by the advanced bots. It is concluded to be a bot if the majority of the data represents this characteristic while if that is not true it analyzes the key difference time on a similar condition. If it results in zero or near zero then it is a bot. It is not feasible that the time between pressing different keys is always or near zero. The rest cases, go through further analysis. Next, it checks if a bot is providing a scripted time delay in key presses. Time delay refers to deliberate time added after pressing or releasing each key. It can be constant or variable. This delay would evade our previous checkers easily. For example, a bot adds a time delay for each key press in the pattern for key Time Hold as 0.25, 0.5, 0.75, 1, 1.25, 1.5 .... and Key Difference Time as 0.005, 0.010, 0.015, 0.020 ....

As seen above there is an intentional variable time delay added by the bot. This would make it difficult to detect it. So, it then further checks for any pattern that can be seen in both the Key hold time and key differences. Firstly, it checks for the presence of an arithmetic progression series, if it fails then it detects for geometric progression series or lastly for harmonic series. If any of the cases is found to be true it is concluded to be a bot. It provides an additional time range for each pattern check to adjust with the hardware delay to prevent the bot from remaining undetected. If it fails to find a pattern then the user is concluded to be a human.

Case 2, which is followed when the time taken is more than one second. In this case, it checks the presence of Key Hold time and key difference time data. As mentioned in case 1. If it results in True, then it is concluded to be a bot. If it is False then it moves further and checks for any sequence or pattern which can

be detected in both of the lists. If any of them comes out to be True, it is determined to be a bot. If it is False then it checks for the usage of the paste command. If paste was initiated then it checks for its authenticity. For verifying the paste command, all pressed keys go through filtration where it internally calculates and carries out all the functionality of digraph keys that can affect the input like "ctrl+x", which is used to cut the text. It removes other digraph keys that are not affecting the input characters like "ctrl+a", it is used to select the entire text but does not change the input field. It carries out the same procedure for other monograph keys like backspaces, deletes, and special characters. After processing and accordingly modifying the timestamps in key hold time and key difference time. It checks an increase in the number of characters that were present before or after the paste command is given. If there is none then the command is empty while if there is an increase in characters then paste is implemented. Hence, it is authentic similar to case 1. If a paste is not implemented then the input is concluded to be provided by a human.

Standard equation used to detect Arithmetic Progression is as follows [23]

$$T_n = a + (n - 1) * d \quad (1)$$

T_n=Nth term of the sequence a=First term of the sequence d=Common difference in the series

Standard equation used to detect Geometric Progression is as follows [23]

$$A_n = a * r_{(n-1)} \quad (2)$$

A_n=Nth term of the sequence a=First term of the sequence r=Common ratio of the series

Standard equation used to detect Harmonic Progression is as follows [23]

$$T_n = \frac{1}{[a + (n - 1) * d]} \quad (3)$$

T_n= Nth term of the sequence a=First term of the series d=common difference between consecutive terms

4 Result

To begin the experiment it takes an open access dataset which is a Behavioral Biometrics Multi Device and Multi Activity Data From Same Users Dataset [24]. This dataset contains data from 117 users using various devices. This algorithm uses a desktop typing dataset where the user types for approximately 50 minutes on a desktop. Average keystrokes as per the dataset document per user is 11760. Each file contains data of a new user. To run the dataset, it needs to be modified and processed in such a way that it can be run on the designed algorithm. As the data contains various irregularities and only keystroke information that means it does not get the final input written by the user. The first step is data cleaning which includes changing the timestamp such that it can be used in the algorithm. In the data set it could see some irregularities present. For example, the presence of only key presses and no key release for several consecutive keys. Various vice versa situations were also observed where only key release was recorded. Digraph keys were found mixed between these irregularities or were stored similar to these data irregularities. For example, alternate key press time and key release time, so two consecutive key presses and then two consecutive key release data are found which is not a data irregularity but is stored similarly to it. As it can be observed in table 1. So, it works on removing data irregularities present in each Excel file

Proceedings DOI: 10.21467/proceedings.178
ISBN: 978-81-984081-8-1

Series: AIJR Proceedings
ISSN: 2582-3922

and keeping the required data intact. The algorithm proceeds by checking the key press (which is shown as 0) and key release (which is shown as 1) sequence and removing all consecutive keys showing 0 or 1 more than twice together. It updates the rest of the key names and timestamps accordingly. In the data, certain cases are left where there is only one irregular record or when a key is pressed but its release information is present after two or 3 data. So, it works on the information of each key when it is pressed or released and in which order. With this, it removes single irregularities and gets the position of each key's press and release position in the list. Data cleaning is finished and related required data is modified. It creates a new list that stores the key press timestamp detail and another that stores the key release

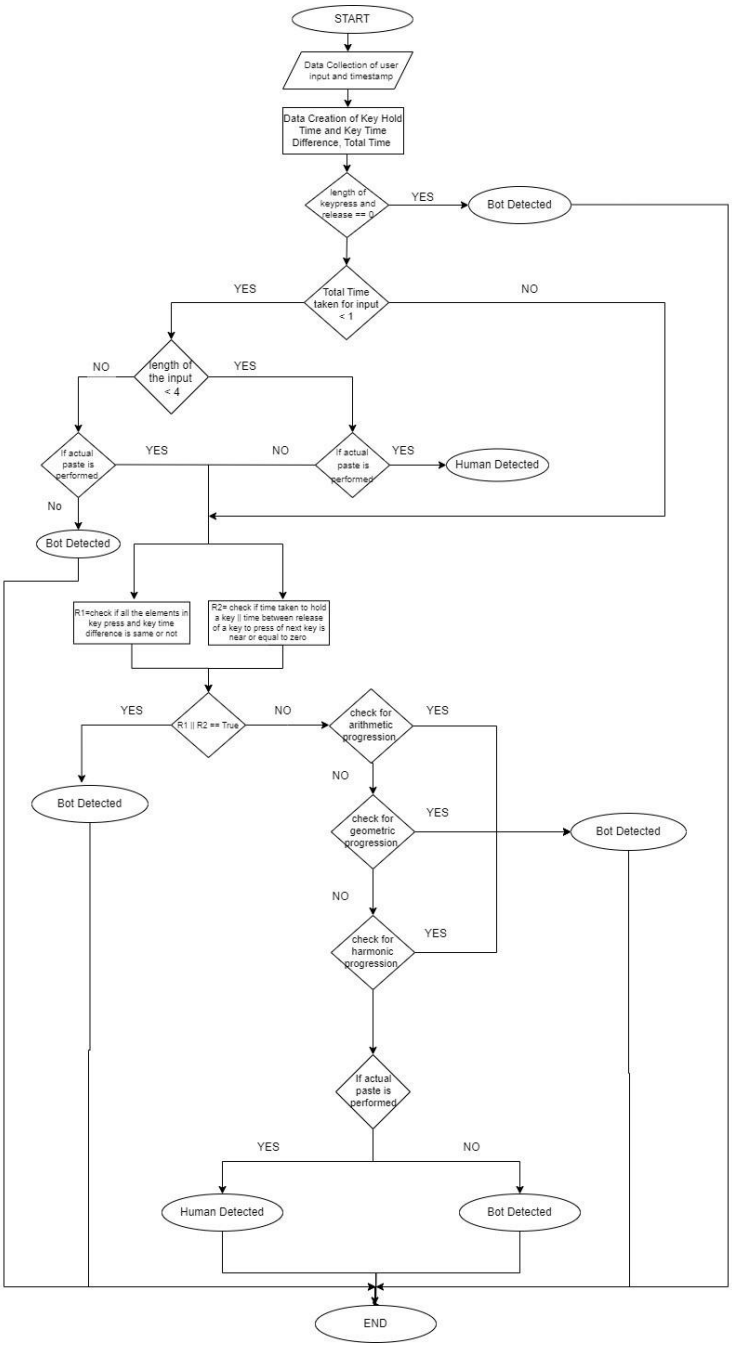

Figure 2. Flowchart Of the overall working of the Algorithm

timestamp for each key present. Hence, now the data is compatible and can be run on the bot detection model which was explained in the earlier section.

Table 1. Data Irregularities

KEY	KEY INFORMATION
BACKSPACE	0
CTRL	0
V	0
V	1
CTRL	0
DELETE	1
DELETE	1
J	0
J	1

0 : KEY PRESS
1 : KEY RELEASE

It starts by providing the entire single file of each unique user to the bot detection model. After some iterations and storing the results it divided the single file into two parts and provided that as input to the model and get separate results for both cases. This is carried out due to the same keystroke features used in the entire dataset as mentioned in the dataset details. So this procedure is iterated multiple times by providing different files and dividing them into any one of two, four, five, ten, twenty, fifty, five hundred equal halves and providing each part as an independent input to the model. This way it verifies the system more than a thousand times. Based on this it detects how correctly it determines. Then to check if it can detect a bot, it makes the bot execute the same file. To execute this similar to the previous case it goes for much smaller parts and lets the bot select a random length string from the entire file and give that as input. For this to work more data processing needs to be done as a bot would write special characters and digraphs instead of enacting them. For example, if backspace is written in the key, it would write backspace instead of following the action of backspace. So, a functionality is added where when it reaches a special character or digraph, it follows those commands instead of inputting them as a word. This includes ctrl+x, ctrl+v, ctrl+c, ctrl+a, delete, backspace, caps lock, shift+character, space, etc. As it can be seen in 4 here it gives space after reading the space key written and removes L, T characters after reading the keys which require actions. It types the rest of the keys as it is which do not require any action. Hence, converting the list of the written key information into an executable form.

The bot selects a random string and types while differentiating between keys to type and actions to follow. After that, it follows the same procedure to identify whether it is a bot or not as mentioned in the original code. Adding another

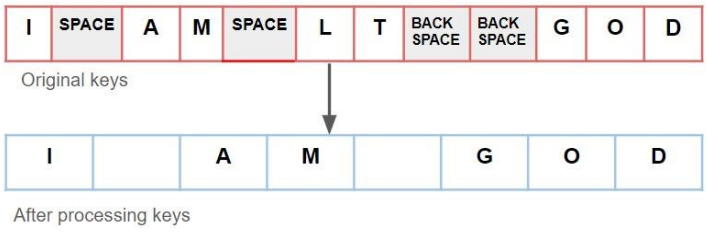

Fig.4. Data processed for bot to provide input

Proceedings DOI: 10.21467/proceedings.178
ISBN: 978-81-984081-8-1

Series: AIJR Proceedings
ISSN: 2582-3922

difficulty an intentional time delay is added in either key hold time or key time difference or even both in some cases to improve the bot level making bot detection difficult. For this, it picks new files and then extracts random strings with different time delays that can be constant throughout the string. It iterates this with different time delays for a variety of the users present. Next, it adds a time delay in a pattern sequence. For example, it adds different time delays for each keystroke using a common difference making it an arithmetic progression. For some cases, time delay is added with a common ratio making it a geometric progression or even making time delay in harmonic progression. To make the bot more advanced instead of adding a time delay in a pattern series that can be detected, it selects a random time delay for each keystroke and uses that for that particular case. This adds to the randomness of key hold and key difference time list making it more difficult to detect if it is a bot or not.

The result is then added based on accuracy. It creates a confusion matrix for both of the cases mentioned above. The first case refers to the direct input from the file, the second case refers to the usage of repeated data mentioned in the dataset. It combines the data from both cases and gets an accuracy of 0.9755 for our bot detection model.

5 Conclusion and Future Scope

As technology continues to advance, bots are becoming increasingly sophisticated, developing the ability to bypass various detection methods with ease. This growing challenge highlights the urgent need to strengthen web safety and security measures to protect against malicious activities. One of the significant risks posed by bots is their capacity to perform fraudulent transactions, scrape valuable data, or overload servers with requests, leading to significant disruptions. To address these issues, this dissertation proposes a novel approach to bot detection by utilizing keyboard behavioral analysis, offering an additional layer of security alongside existing detection techniques. The framework introduced in this study focuses on analyzing user input, particularly the patterns of key presses and timing, and then comparing these behaviors with known characteristics of human and bot interactions. Human users typically exhibit unique typing rhythms, such as varying key hold times and pauses between keystrokes, which bots struggle to mimic accurately. By recognizing these differences, the proposed methodology can accurately detect and distinguish between human and bot users, providing a high level of security for web pages. This approach not only enhances the detection accuracy but also promotes resource efficiency, as it requires minimal processing power and does not place additional strain on server resources. Moreover, it helps reduce bandwidth usage by preventing unnecessary interactions from malicious bots. As a result, this method significantly contributes to a safer and more secure web environment, protecting websites from a wide range of bot-driven threats. The result of the bot detection mechanism includes accuracy of 0.9755.

In future iterations, the algorithm can be further enhanced by integrating advanced pattern recognition capabilities. These improvements could include identifying even more subtle patterns in key hold times and key time differences, thereby increasing the model's precision and efficiency, making it even more effective in combating evolving bot strategies.

6 Declarations

6.1 Competing Interests

The author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

6.2 Publisher's Note

AIJR remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

How to Cite

Vasvi Sharma, Siddhartha Chauhan (2025). Web Bot Detection Using Keyboard Behavioural Analysis. *AIJR Proceedings*, 79-90. https://doi.org/10.21467/proceedings.178.10

References

[1] G. Stragapede, R. Vera-Rodriguez, R. Tolosana, A. Morales, N. Damer, J. Fierrez, and J. Ortega-Garcia, "Keystroke verification challenge (kvc): Biometric and fairness benchmark evaluation," *IEEE Access, vol. 12*, pp. 1102–1116, 2024.

[2] "What is content scraping? | web scraping." [Online]. Available: https://www.cloudflare.com/learning/bots/what-is-content-scraping/

[3] A. A. Wahab, D. Hou, and S. Schuckers, "A user study of keystroke dynamics as second factor in web mfa," in *Proceedings of the Thirteenth ACM Conference on Data and Application Security and Privacy, ser. CODASPY '23*. New York, NY,USA: Association for Computing Machinery, 2023, p. 61–72. [Online]. Available:https://doi.org/10.1145/3577923.3583642

[4] D. Rudrapal, S. Das, and A. Saha, "Analysis of user keystroke dynamics and its application for preventing automated form filling programmes," in *International Conference on Computing Communication and Information Technology, vol. 27*,2012, p. 176.40

[5] G. Suchacka, A. Cabri, S. Rovetta, and F. Masulli, "Efficient on-the-fly web bot detection," *Knowledge-Based Systems*, vol. 223, p. 107074, 2021.

[6] F. Monrose and A. D. Rubin, "Keystroke dynamics as a biometric for authentication*," Future Generation computer systems*, vol. 16, no. 4, pp. 351–359, 2000.

[7] "What is web indexing?" [Online]. Available: https://www.simpletiger.com/resources/glossary/web-indexing

[8] P. Laperdrix, N. Bielova, B. Baudry, and G. Avoine, "Browser fingerprinting: A survey," *ACM Transactions on the Web (TWEB)*, vol. 14, no. 2, pp. 1–33, 2020.

[9] E. Bursztein, S. Bethard, C. Fabry, J. C. Mitchell, and D. Jurafsky, "How good are humans at solving captchas? a large scale evaluation," in 2010 *IEEE Symposium on Security and Privacy*, 2010, pp. 399–413.

[10] "What is a web crawler? | how web spiders work." [Online]. Available: https://www.cloudflare.com/learning/bots/what-is-a-web-crawler/

[11] S.-S. Shen, S.-H. Lin, T.-H. Kang, and W. Chien, "Enhanced keystroke dynamics authentication utilizing pressure detection," in 2016 *International Conference on Applied System Innovation (ICASI). IEEE, 2016,* pp. 1–4.

[12] H.-j. Lee and S. Cho, "Retraining a keystroke dynamics-based authenticator with impostor patterns," *Computers & Security, vol. 26,* no. 4, pp. 300–310, 2007.

[13] Z. Chu, S. Gianvecchio, A. Koehl, H. Wang, and S. Jajodia, "Blog or block: Detecting blog bots through behavioral biometrics," *Computer Networks, vol. 57*, no. 3, pp. 634–646, 2013.

[14] K. Thomas, D. McCoy, C. Grier, A. Kolcz, and V. Paxson, "{Trafficking} fraudulent accounts: The role of the underground market in twitter spam and abuse," in *22nd USENIX Security Symposium (USENIX Security 13)*, 2013, pp. 195–210.

[15] X. Lu, S. Zhang, P. Hui, and P. Lio, "Continuous authentication by free-text keystroke based on cnn and rnn," *Computers & Security, vol. 96*, p. 101861, 2020.

[16] K. A. Rahman, D. Neupane, A. Zaiter, and M. S. Hossain, "Web user authentication using chosen word keystroke dynamics," in 2019 *18th IEEE International Conference On Machine Learning And Applications (ICMLA). IEEE, 2019*, pp.1130–1135.

[17] D. Migdal and C. Rosenberger, "Statistical modeling of keystroke dynamics samples for the generation of synthetic datasets," *Future Generation Computer Systems,vol. 100*, pp. 907–920, 2019.

[18] D. DeAlcala, A. Morales, R. Tolosana, A. Acien, J. Fierrez, S. Hernandez, M. A.Ferrer, and M. Diaz, "Becaptcha-type: Biometric keystroke data generation for improved bot detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023*, pp. 1051–1060.

[19] R. U. Rahman and D. S. Tomar, "New biostatistics features for detecting web bot activity on web applications," *Computers & Security, vol. 97*, p. 102001, 2020.

[20] T. Neacsu, T. Poncu, S. Ruseti, and M. Dascalu, "Doublestrokenet: Bigram-level keystroke authentication," *Electronics, vol. 12*, no. 20, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/20/4309

[21] K. Revett, "A bioinformatics based approach to user authentication via keystroke dynamics," *International Journal of Control, Automation and Systems, vol. 7*, pp.7–15, 2009.

[22] C. Iliou, T. Kostoulas, T. Tsikrika, V. Katos, S. Vrochidis, and I. Kompatsiaris, "Detection of advanced web bots by combining web logs with mouse behavioural biometrics," *Digital threats: research and practice*, vol. 2, no. 3, pp. 1–26, 2021.

[23] D. Canali and D. Balzarotti, "Behind the scenes of online attacks: an analysis of exploitation behaviors on the web," in *20th Annual Network & Distributed System Security Symposium* (NDSS 2013), 2013, pp. n–a.

[24] B. Amin Azad, O. Starov, P. Laperdrix, and N. Nikiforakis, "Web runner 2049: Evaluating third-party anti-bot services," in *Detection of Intrusions and Malware,and Vulnerability Assessment: 17th International Conference, DIMVA 2020*, Lisbon,Portugal, June 24–26, 2020, Proceedings 17. Springer, 2020, pp. 135–159.