Detection of Advanced Web Bots by Combining Web Logs with Mouse Behavioural Biometrics

CHRISTOS ILIOU, Information Technologies Institute, CERTH and BU-CERT, Bournemouth University
THEODOROS KOSTOULAS, Department of Information & Communication Systems Engineering, University of the Aegean and Department of Computing and Informatics, Bournemouth University
THEODORA TSIKRIKA, Information Technologies Institute, CERTH
VASILIS KATOS (https://orcid.org/0000-0001-6132-3004), BU-CERT, Bournemouth University
STEFANOS VROCHIDIS and IOANNIS KOMPATSIARIS, Information Technologies Institute, CERTH

Digit. Threat.: Res. Pract., Vol. 2, No. 3, Article 24, Publication date: June 2021.
DOI: https://doi.org/10.1145/3447815 (https://doi.org/10.1145/3447815)

Web bots vary in sophistication based on their purpose, ranging from simple automated scripts to advanced web bots that have a browser fingerprint, support the main browser functionalities, and exhibit a humanlike behaviour. Advanced web bots are especially appealing to malicious web bot creators, due to their browserlike fingerprint and humanlike behaviour that reduce their detectability. This work proposes a web bot detection framework that comprises two detection modules: (i) a detection module that utilises web logs, and (ii) a detection module that leverages mouse movements. The framework combines the results of each module in a novel way to capture the different temporal characteristics of the web logs and the mouse movements, as well as the spatial characteristics of the mouse movements. We assess its effectiveness on web bots of two levels of evasiveness: (a) *moderate web bots* that have a browser fingerprint and (b) *advanced web bots* that have a browser fingerprint and also exhibit a humanlike behaviour. We show that combining web logs with visitors' mouse movements is more effective and robust toward detecting advanced web bots that try to evade detection, as opposed to using only one of those approaches.

CCS Concepts: • Information systems → Web log analysis; Traffic analysis; **• Computing methodologies → Supervised learning by classification**;

Additional Key Words and Phrases: Web bot detection, evasive web bots, advanced web bots, mouse movements, mouse biometrics, humanlike behaviour

ACM Reference format:
Christos Iliou, Theodoros Kostoulas, Theodora Tsikrika, Vasilis Katos, Stefanos Vrochidis, and Ioannis Kompatsiaris. 2021. Detection of Advanced Web Bots by Combining Web Logs with Mouse Behavioural Biometrics. *Digit. Threat.: Res. Pract.* 2, 3, Article 24 (June 2021), 26 pages. https://doi.org/10.1145/3447815 (https://doi.org/10.1145/3447815)

1 INTRODUCTION

Web bots are an integral part of the web, since they allow the automation of several vital tasks, some of which would have otherwise been impossible to perform. They are responsible for numerous browsing automation processes, such as web indexing, website monitoring (validation of hyperlinks and HTML code), data extraction for commercial purposes, and feed fetching web content. Some of these tasks require web bots to visit web servers repeatedly and, in some cases, for prolonged periods of time. As a result, web bots generate a huge amount of web traffic; based on Distil Networks' 2019 report [15], web bots accounted for the 37.9% of the total traffic that they monitored.

Based on their purpose, web bots vary in sophistication [14, 15]. For example, to download the HTTP content of a web server, a simple web bot would be sufficient. However, when it is required to test a web server's functionality by filing web forms, running complex JavaScript code on a web page, and performing mouse movements (i.e., mouseover actions to specific elements of a web page), then a web bot that supports all functionalities of a web browser should be used.

Such web bots may though be abused for malicious purposes. These malicious web bots can perform highly complex tasks and at the same time try to avoid detection by presenting a browser fingerprint and a humanlike behaviour [14, 15, 20]. This makes them particularly dangerous, since they present themselves as humans and perform several actions in a humanlike way, which severely hinders their detectability. Examples of malicious behaviours of such web bots include trying out different credit card numbers, gift card numbers, and login credentials, buying all the available stock of specific limited products to later resell at higher price (i.e., scalper bots), holding items in shopping carts thus preventing access to valid customers, scraping item prices to gain competitive advantage, and generating accounts to spam messages or amplify propaganda [15].

Due to the dangers associated with web bot visitors, it is in the best interest of web servers to place special restrictions on web bots upon detection, so that bots cannot perform malicious acts. To detect web bots, current state-of-the-art approaches both in academia [18, 19, 43] and in commercial solutions [1, 15] propose, besides the rule-based web bot detection techniques, the use of machine learning to distinguish bots from human visitors. In machine learning-based techniques, past web sessions are used to train models of the human and web bot behaviour; these models are then used to classify new visitors as web bots or humans.

Current research on machine learning-based web bot detection uses either web logs [3, 12, 13, 29, 34, 37, 38] or mouse movements [12, 42] to detect web bots. Even though the aforementioned techniques are highly accurate, they do not address one key aspect of the problem, which is that web bots might try to exhibit both a browser like fingerprint and a humanlike behaviour to avoid detection [15, 20].

In this research, we propose a web bot detection framework to detect malicious web bots of different sophistication levels that try to evade detection. The framework combines two web bot detection modules, one that relies on web logs and one that leverages mouse behavioural biometrics (i.e., mouse movements). The rationale behind the combination of the two detection modules is to capture the different temporal characteristics of the web logs and the mouse movements, as well as the spatial characteristics of the mouse movements, to make a more robust detection framework with the goal to make it even harder for web bots to evade detection. To evaluate our framework, we performed experiments by using two types of web bots with different sophistication levels: (i) the *moderate web bots* that have a browser fingerprint and (ii) the *advanced web bots* that have a browser fingerprint and exhibit a humanlike behaviour.

This work uses as a basis and expands the framework proposed in Reference [20] and addresses the challenges that the authors outlined. More specifically, to overcome the limitations of the web bot detection framework in Reference [20] in detecting advanced web bots based on features extracted from web logs, this framework (i) introduces an additional module that detects web bots based on their mouse movements, and (ii) proposes a novel way of combining the detection modules utilised (i.e., the one that uses web logs and the one that uses mouse movements) to enable capturing the different temporal characteristics of the web logs and the mouse movements, as well as the spatial characteristics of the mouse movements.

Additionally, in this research, we further split the web bots that were defined as "advanced" in Reference [20] into *moderate web* bots and *advanced web bots*, both of which present a browser fingerprint but otherwise have different levels of sophistication; this differs from Reference [20] where web bots were split based on whether they present a browser fingerprint or not. This work does not consider the simple web bots (i.e., simple scripts) of Reference [20], since they have a botlike fingerprint and perform no mouse movements, hence they can be detected in a deterministic way. The further categorisation of the web bots referred to as "advanced" in Reference [20] into "moderate" and "advanced" web bots in this work, on the basis of their evasiveness, allows us to further showcase how the increase in their evasiveness capabilities affects the performance of our detection models. The behaviour and configuration of the tested web bots is derived from techniques that have been proposed in literature for evasive web content gathering [11, 21, 27] in combination with techniques that stem from the observation of advanced web bots in the wild [14, 15].

The contributions of this article are as follows:

- A novel machine learning-based web bot detection framework that combines web logs and mouse movements to detect advanced web bots in a novel way by capturing the different temporal and spatial characteristics of the types of data utilised. This results in a more robust detection framework, which is able to detect web bots even in cases where one of the individual detection modules fails.
- A methodology for evaluating the proposed framework, which takes into account the different levels of evasiveness that web bots may exhibit. As opposed to previous research where only simple scripts vs. advanced web bots were considered, in this work we investigate how increasing the evasiveness of web bots affects the performance of our framework.

The rest of this article is organised as follows: Section 2 provides the background on the web bot detection problem and covers the related work. Section 3 defines our threat model. Section 4 presents an overview of our framework while Section 5 presents the classification methods utilised in the framework. Section 6 describes the evaluation methodology and the experimental setup, while Section 7 presents the evaluation results. Section 8 analyses and interprets the findings and the limitations of our work. Finally, Section 9 summarises our work and outlines future work.

2 BACKGROUND AND RELATED WORK

Web bot detection aims to accurately distinguish whether a web visitor is a bot or a human. This categorisation may entail simply distinguishing web bots from human visitors [3, 4, 10, 12, 18, 34] or further categorising web bots based on their functionality [17], purpose [6, 32, 43], or complexity (i.e., simple vs. sophisticated) and based on whether they try to evade detection or not [20].

Early versions of web bots were simple scripts [14]. These bots were easy to detect by comparing their fingerprints with the ones of common browsers. Such fingerprints can include the headers of the visitor's requests, whether they support JavaScript, cookies, and web sessions, values of specific JavaScript variables, and so on. However, the introduction of browsing automation tools, such as Selenium,[1] enabled the effortless creation of more sophisticated web bots that can support the majority of the features that common browsers offer as well as allow the bots to interact with the server as they were humans (i.e., perform mouse movements, fill in forms, click elements, etc.), which makes their detection more challenging. Even though such web bots, in their vanilla configurations, contain fingerprints that can reveal their bot nature [23], additional configurations can be applied to avoid detection [22, 23]. Moreover, web bots can be designed to use the regular browsers of a machine (instead of using an automated browsing software), which makes fingerprint-based detection even harder [2].

Since there is no universal definition for the different types of web bots based on their detection evasion capability, we categorise the web bots into three types, (i) the *simple web bots*, (ii) the *moderate web bots*, and (iii) the *advanced web bots*. The *simple web bots* are simple scripts that have neither a browserlike fingerprint nor a humanlike behaviour. The *moderate web bots* have a fingerprint that is close to one of a browser (and, for the purposes of this research, we assume that we cannot distinguish them by their fingerprint), support the main functionalities of a browser, but they do not present any humanlike browsing behaviour. Thus, moderate web bots perform mouse movements and click on hyperlinks but they do not exhibit any humanlike (i.e., intelligent) behaviour: their mouse movements traverse between hyperlinks in a straight line, and they follow hyperlinks randomly without waiting between consecutive requests. Finally, the *advanced web bots* are web bots that have a browser like fingerprint (similar to the one of the moderate web bots), and also exhibit a humanlike browsing behaviour. Based on Distil Networks' 2019 report [15], simple web bots accounted for 26.4% of the total web bot traffic, more complex web bots that use “headless browsers” software accounted for 52.5%, and sophisticated web bots (which are in principle similar to the ones we call advanced web bots) accounted for 21.1%.

For many years, the most popular techniques for detecting web bots were based on **Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHAs)** [40]. CAPTCHA challenges are usually based on visual challenges that can be accompanied with aural ones for the visually impaired. The tests are based on the assumption that a human can easily fulfil these visual challenges, while a web bot cannot. There are multiple CAPTCHA-like challenges, examples of which include the click of a checkbox on a web page [36], the selection of images that fulfil certain criteria (e.g., Google's reCAPTCHA v2[2]), and the extraction of letters from a distorted image or audio file.

While visual/aural-based CAPTCHA challenges used to be effective for the detection of web bots, current advances in image processing and speech recognition have reduced their effectiveness. A variety of highly accurate techniques have been proposed to bypass popular CAPTCHA challenges, ranging from simply using public online speech to text engines [9], to a combination of several techniques, including image reverse search, tagging, recognition, and processing [35]. These attacks led CAPTCHA challenges to increase in difficulty. Original versions of CAPTCHA challenges have received a lot of criticism, especially from people with disabilities who sometimes struggle with fulfilling these requests and also from people who feel that their everyday work is slowed down.

The usability and effectiveness issues associated with visual challenges led current research to focus on rule-based and behavioural-based detection techniques that do not affect the user experience (i.e., they do not interrupt the user to ask them to solve some visual challenges unless an abnormal behaviour has been detected [15]). Additionally, the latest version of Google's CAPTCHA challenge[3] (version 3), introduced at the end of 2018, also performs adaptive risk analysis based on the context of the action and returns a score for each request without user friction.

Current web bot detection approaches used by commercial solutions advertise that they combine (i) rule-based web bot detection based on browser fingerprinting techniques, as well as (ii) web bot detection based on the behaviour of the visitors (e.g., the spatial characteristics of the mouse movements, the browsing speed, etc.) [15]. After a visitor is identified as a bot with the aforementioned approaches, additional steps are taken (which can also be chosen by the administrators of the sites) such as to block the visitor, deliver different content, or request from the visitor to prove that they are human by solving some visual challenges [15].

Rule-based web bot detection techniques that use browser fingerprints include font detection, plugin enumeration, WebGL fingerprinting, examination of unique to browser automation software strings in JavaScript variables, and more [5, 23, 31]. Furthermore, more advanced fingerprinting techniques have been proposed that can extract low level properties, such as the instruction-set architecture, and the used memory allocator [31]. However, research has shown that current commercial tools that detect web bots based on their fingerprint present several flaws [5]. For example, some of those techniques are unique to specific automation tools and even specific versions of the tools, making the preservation of those fingerprint lists time-consuming [5, 24]. Additionally, such techniques can be evaded if the fingerprints of the automation tools are changed [5, 39] or by utilising regular browsers instead of browsing automation software [2, 14].

Regarding the detection of web bots based on their behaviour, the most prominent approach is the use of machine learning and, more specifically, the use of classification [20, 34, 37] or clustering algorithms [3, 13, 38]. Furthermore, detection algorithms proposed in literature use either web logs [3, 12, 13, 29, 34, 37, 38] or mouse movements [12, 42] to detect web bots. Additionally, the detection process can be performed either offline (i.e., decide after the end of a session whether it is from a bot), or online by performing an estimation during the session [12, 29].

The web bot detection approaches that are based on web logs rely primarily on several “traditional” machine learning algorithms, such as Support Vector Machines [20, 29, 37], Random Forests [20, 34], Adaboost [20, 34], and Multi-layer Perceptron classifiers [10, 20, 29, 37]. Initially, the sessions of each visitor of the web server are extracted from the web logs [7, 20, 26, 30, 33, 34]. Then, for each session, several measurable properties/characteristics (features) of the visitors' behaviour are calculated, including the access frequency of web pages [20, 38], the type of web content accessed (e.g., HTML, text, JavaScript, image, css, etc.) [18, 20, 30], the access patterns [4, 20], and the HTTP errors produced [7, 20, 38]. The calculated feature values are used as input to train machine learning models and the trained models are then used to classify the new visitors as bots or humans.

Latest research has introduced the use of mouse movements for the detection of web bots [12, 42]. Mouse movements can either be represented as images and used directly as input to **Convolutional Neural Networks (CNNs)** [42] or have several high level actions extracted from them, such as click, point-and-click, and drag-and-drop [12]. For the latter, these actions accompanied with their properties, such as the distance of the mouse move, its duration, and efficiency, are used to train classification models. Table 1 shows the performance of works that have been proposed in literature. It is shown that the more sophisticated the web bots are, the more difficult it is to detect them.

Table 1. Performance of Web Bot Detection Frameworks in Literature

Authors	Input	Type of bots	Results
Rovetta et al. [29]	web logs	mix of web bots, majority should be simple	accuracy=0.98
Stevanovic et al. [37]	web logs	simple	accuracy>0.99, F-score>0.80 (imbalanced classes)
Cabri et al. [10]	web logs	simple	F-score>0.955
Iliou et al. [20]	web logs	simple	precision, recall, and balanced accuracy >0.95
		moderate and advanced	balanced accuracy=∼0.55
Chu et al. [12]	mouse movements	between moderate and advanced	TPR=0.63∼0.99 (based on the bots' complexity), TNR>0.99
Wei et al. [42]	mouse movements	between moderate and advanced	TNR=0.93∼0.99 (based on the bots' complexity)

Although the aforementioned techniques show promising results, they do not address a key aspect of the web bot detection problem, which is the identification of web bots that try to evade detection via, for example, exhibiting a browser fingerprint and a humanlike behaviour [11, 20, 21, 27]. Based on the current detection techniques, advanced web bots could be modelled by adversaries by (i) having a browser fingerprint, and (ii) exhibiting a humanlike behaviour. A fingerprint that is close to one of a browser can be be achieved with the use and configuration of specific browsing automation software [11, 27, 35], or by using regular browsers [2]. A humanlike behaviour can be achieved by time sleeps that take into account the length of text in each web page [11, 27], crawling web pages at a specific time (e.g., day/night) [11, 27], skipping or spending a few seconds on content that they have already visited [11], and performing human assisted [27] or automatic [11] logging in to websites. Recent research has shown that, although detecting simple bots is relatively easy, detecting more sophisticated web bots that use a browser fingerprint and/or exhibit a humanlike behaviour, including the performance of mouse movements, is much more difficult [20].

To address this problem, in this research, we propose a machine learning-based web bot detection framework that combines web logs and mouse movements for the detection of advanced web bots. This framework can be combined with rule-based approaches to increase the performance of the web bot detection tools. This work uses as its basis the framework proposed in Reference [20] and the respective challenges identified in detecting advanced web bots, and proposes a framework that can be used to effectively detect such bots. Specifically, the proposed framework combines two detection modules: (i) a detection module that uses web logs and (ii) a detection module that uses mouse movements. The module that uses web logs extracts the most promising features for web bot detection as they have been identified in the literature and creates an ensemble classifier that combines several well-established classifiers that have shown promising results in the web bot detection problem [20]. The second module uses the mouse movements of visitors on each web page to train models that can identify web bot and human mouse movement behaviour [42]. The two modules are combined in a novel way to enable the capturing the different temporal characteristics of the web logs and the mouse movements, as well as the spatial characteristics of the mouse movements.

3 THREAT MODEL

For the purposes of this research, we consider the potentially vulnerable assets to be web servers that host content that is of value to specific actors and would therefore motivate them to deploy (malicious) web bots for harvesting its contents and perform activities such as web scraping, competitive data mining, personal and financial data harvesting, and more. This is in line with reports of Distil Networks [14, 15], an industry leader in the bot detection domain, on the typical uses of such (malicious) web bots; specifically, these reports identify valuable content harvesting (e.g., proprietary content scraping and price scraping) among the important challenges for several business sectors.

Such web bots can crawl web servers in a humanlike manner to collect information making them harder to detect. Additionally, we can assume that the malicious web bots exhibit a fingerprint that is indistinguishable from that of a browser as in the opposite case, such bots could be deterministically detected using advanced fingerprinting techniques [5, 22, 23, 31]. This is a logical assumption, as the respective Indicators of Compromise have a low pain threshold (i.e., they require low effort to be changed) [8]. Thus, evasion by advanced threat actors is trivial through the use of the automation tools allowing configurable fingerprints [5, 39] or by utilising regular browsers instead of browsing automation software [2].

Overall, we consider our adversaries to be web bots that have a fingerprint that is indistinguishable from that of a browser and also exhibit a humanlike behaviour; in this research we refer to these as *advanced web bots*. Nevertheless, for comparison purposes and as not all web bot visitors exhibit advanced—according to our definition—behaviour, we also utilised web bots that have a browser fingerprint, support the main functionalities of a browser, but do not exhibit a humanlike behaviour.

4 WEB BOT DETECTION FRAMEWORK

In this section we present the web bot detection framework that uses both web logs and mouse behavioural biometrics for the detection of advanced web bots. To capture the different temporal and spatial characteristics of the data, the framework utilises two detection modules: (i) a module that detects web bots from web logs and (ii) a module that detects web bots from mouse movements. Each of the modules builds its own classifier. The first module builds a classifier that aims to identify whether a visitor corresponds to a bot or a human based on features such as the access frequency of web pages, the type of web content accessed, the access patterns, and the HTTP errors produced [20, 34, 37]. The second module builds a classifier that uses features extracted from the visitors' mouse movements to detect web bots [12, 42].

One of the reasons behind using two different detection modules (and thus two different classifiers) instead of performing a feature level fusion is the complementarity between the two modules based on the different levels of granularity that they provide. This allows us to model the different temporal and spatial characteristics of the visitors' browsing behaviour, including their behaviour regarding the web pages visited, as well as the mouse movements performed in each page.

The general architecture of the framework is presented in Figure 1. The framework uses a database that contains the web logs of each user, as well as the respective mouse movements of each user on each web page. The web logs are used as input to the "web log" detection module, while the mouse movements are used as input to the "mouse movements" detection module. Each module assigns a score ranging from 0 to 1 to each visitor. A high score means that a visitor is very likely to be a bot and a low score that the visitor is very likely to be a human. The respective scores from the two modules are combined to decide whether the session is performed by a bot or a human, depending on whether the resulting score is above a threshold value.

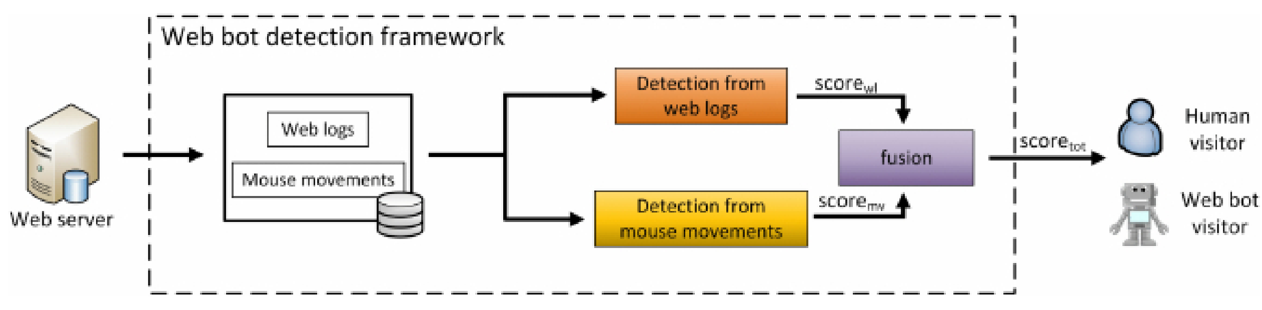


Fig. 1. Web bot detection framework architecture.

Next, we describe in detail the detection module that uses web logs (Section 4.1), the detection module that uses mouse movements (Section 4.2), and the fusion process (Section 4.3).

4.1 Detection from Web Logs

The module that uses web logs to detect web bots is derived from the most prevalent relevant techniques that have been proposed in literature [18, 19, 20, 43]. The architecture of this module is shown in Figure 2. The framework uses a regular expression to parse the web server logs. This allows the trivial application of the framework to any web server logs, provided that they contain all the necessary information.

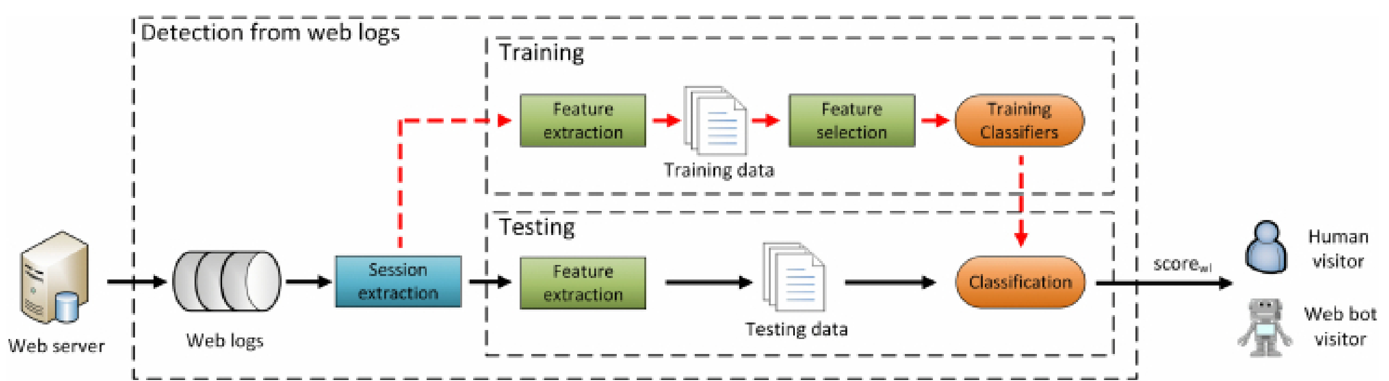


Fig. 2. Web bot detection module that uses web logs.

After the successful connection of the framework with the web server logs, the session extraction procedure takes place. The web log data are split into sessions based on the visitors' PHP session id (Section 4.1.1). For each session, several measurable properties/characteristics (features) of the visitors' behaviour are calculated (Section 4.1.2). Subsequently, the most effective features are selected (Section 4.1.3). The final feature vectors are used to create the classification models (Section 4.1.4). During the testing phase, we feed new data to these classification models to assess their ability to identify web bots. To do that, we initially extract the sessions and feature values from the test data (Sections 4.1.1 and 4.1.2). After that, the most effective features found in the training process are selected (Section 4.1.3). These data are used as input to the (already) trained classifiers to label the new visitors as web bots or humans (Section 4.1.4).

4.1.1 *Session Extraction.* The first step in identifying whether a visitor is a human or a web bot is the extraction of the visitor's session(s) from the log files. Current approaches in literature combine the IP with the browser agent name for the creation of a unique identifier per visitor [7, 18, 19, 20, 30, 34]. This is used primarily when no session ids exist or are logged. However, in our case, we consider that the logging process of the web server is able to store the PHP session id along with the HTTP log data. Thus, we extracted the visitor sessions from the web log files based on their PHP session id. We consider that a session has been completed when more than 30 minutes have passed and no new requests with its id have been performed [7, 18, 19, 30, 34].

4.1.2 *Feature Extraction.* The information included in each session is encoded into measurable values, which are meant to represent the various properties (features) that can be found in (or deduced from) the web logs. These features are related to the method/response code of the HTTP request, the type of file(s) requested, and the browsing behaviour. These features are used as input to train the classification models.

4.1.3 *Feature Selection.* In machine learning, some of the available features might negatively affect the effectiveness of the classification models. Therefore, it is common to perform a feature selection process, in which a subset of all available features is chosen [13, 20]. The framework can be combined with any feature selection algorithm to accommodate different types and volumes of data. The specific details of the feature selection process that has been selected for the evaluation of this framework are presented in Section 6.4.

4.1.4 *Classification.* The final feature vectors are used as input to train the classifiers. The framework supports the construction of an ensemble from several classifiers (i.e., it performs a class probability averaging of all the available classifiers) [20, 34]. The classification algorithms that are typically employed include Support Vector Machines, Random Forest, and so on; details of the classifiers that are employed in this work are presented in Section 5.1. In the testing phase, the ensemble is used to identify whether new web sessions are from web bots or humans.

4.2 Detection from Mouse Movements

The second module of our framework uses mouse movements to identify whether the visitor is a bot or a human, a technique that has been proven to be effective in detecting web bots [12, 42]. The framework constantly collects the mouse movements of each visitor along with the respective timestamps (Section 4.2.1). This information is then used to train classification models for the detection of web bots based on their mouse movements (Section 4.2.2). The architecture of this module is presented in Figure 3.

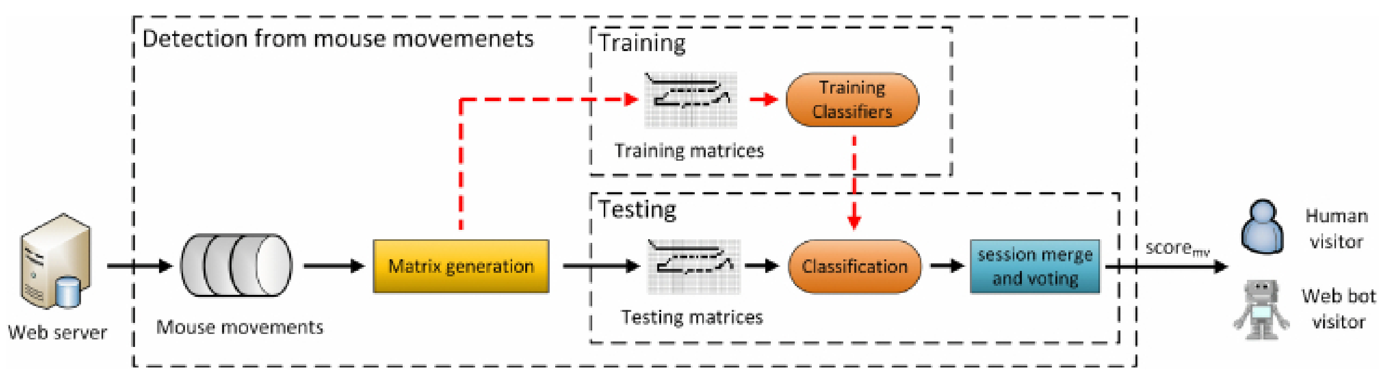


Fig. 3. Web bot detection module that uses mouse movements.

4.2.1 *Mouse Movement Collection Architecture.* The first step is to collect the mouse movements of each visitor on each web page. These data are stored as a sequence and include all the points that the visitor performed mouse movements on, along with the respective timestamps. The data are collected in the form of $\{(x_1, y_1, t_1), (x_2, y_2, t_2), \ldots, (x_n, y_n, t_n)\}$, where x_i and y_i are the coordinates of the current mouse point, t_i is the timestamp of when the mouse move was performed, and n is the total number of points over which the mouse hovered in each web page.

The process for the collection of mouse movements is presented in Figure 4. To enable our framework to collect such data, a JavaScript file is embedded in each web page.[4] This JavaScript file constantly stores locally the mouse movements of the browser along with the respective timestamps and sends them back to the server when the visitor performs a mouse click (left/right/middle), or periodically, every few seconds. The main idea behind this is to track the mouse movements in every web page that has been visited. There are two main ways that a visitor can visit a web page in our setting: (i) either by clicking on a hyperlink in the web page or (ii) by using the browser's general functionality, i.e., writing the URI of the web page or clicking the browser "back" button. In the first case, a mouse click event is triggered, allowing us to get all the currently collected mouse movements. In the second case, to the best of our knowledge, the best (and most browser-software independent) approach is to periodically collect the current mouse movements.

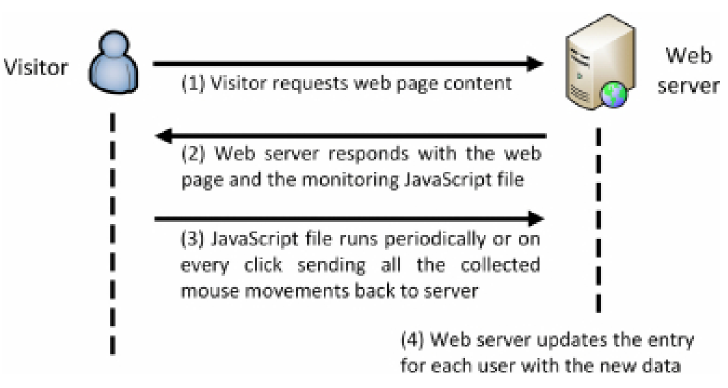

Visitor
Web server
(1) Visitor requests web page content
(2) Web server responds with the web page and the monitoring JavaScript file
(3) JavaScript file runs periodically or on every click sending all the collected mouse movements back to server
(4) Web server updates the entry for each user with the new data

Fig. 4. Mouse movement collection process.

4.2.2 *Classification.* The mouse movements that each visitor performed on each web page are grouped together to form sequences. More specifically, the format of the sequences is $\{(x_1, y_1, dt_1), (x_2, y_2, dt_2), \ldots, (x_{n-1}, y_{n-1}, dt_{n-1})\}$, where x_i and y_i are the coordinates of the current mouse point, $dt_i = t_{i+1} - t_i$ is the difference between the timestamps of when the mouse move was performed on the points i and $i+1$ (i.e., the total time the mouse stayed on the corresponding point) and n is the total number of points over which the mouse hovered in each web page. Since t_n is the current time (and thus t_{n+1} is unknown), the last point in the sequence is $(x_n, y_n, 0)$. Each sequence is presented as a two-dimensional matrix where x_i and y_i correspond to the indexes of each element in the matrix and dt_i corresponds to its value.

Each session creates several two-dimensional matrices, one per page visited. These matrices are used to train classification models for the detection of web bots (by considering each two-dimensional matrix as a new training sample). Due to the nature of the problem (i.e., detecting patterns in mouse trajectories) as well as approaches proposed in literature [42], the classifier of this module uses CNNs; details of the classifiers that are employed in this work are presented in Section 5.2.

In the testing phase, since several matrices correspond to the same visitor, we predicted the class for each matrix separately and then performed a majority voting over all matrices in each session to identify whether these are a bot or a human. Specifically, we calculated the total number of matrices that would be individually classified as matrices generated from web bots and from humans, and labeled the session as a bot session, if the number of bot matrices exceeded the number of human matrices.

4.3 Fusion of Detection Models

The framework performs a decision level fusion to leverage the complementarity between the two modules based on the different levels of granularity that they provide that enables the modelling of the different temporal characteristics of the browsing behaviour and mouse movements. At the end of each session, the scores of the two detection modules are combined to decide whether the session is from a bot or a human.

Due to the complexity of human visitor mouse movements and thus the difficulty in simulating them, we believe that detection based on mouse movements can be less susceptible to evasion. Furthermore, relative research has shown that web log-based detection is not very effective in the case of advanced web bots [20].

For this reason, when the score of the detection module that uses mouse movements is either very high or very low (indicating, with high probability, that the visitor is a web bot or a human, respectively) we only take the score from the mouse movement detection module into account. Otherwise, the scores of two detection modules are combined. The equation to calculate the final score is as follows:

$$score_{tot} = \begin{cases} score_{mv}, & \text{if } score_{mv} \geq thres_h \text{ or } score_{mv} \leq thres_l \\ w_{mv} * score_{mv} + w_{wl} * score_{wl}, & \text{otherwise} \end{cases}, \tag{1}$$

where $score_{tot}$ is the final score, and $score_{mv}$ and $score_{wl}$ are the classification scores for the detection modules that use mouse moves and web logs, respectively. The w_{mv} and w_{wl} represent the weights of the score outputs of the two detection modules with $w_{mv} + w_{wl} = 1$ and the $thres_h$ and $thres_l$ are the threshold values that indicate when the detection using mouse movements is reliable enough to be used on its own. The final decision score, $score_{tot}$, is compared to a predefined threshold to determine whether a visitor is a human or a bot.

5 CLASSIFICATION METHODS FOR WEB BOT DETECTION

This section presents the classification methods employed by the two web bot detection modules utilised in this framework. More specifically, we present the features and the machine learning algorithms utilised by the log-based web bot detection module (Section 5.1), as well as the deep learning architecture employed by the detection module that uses mouse movements (Section 5.2).

5.1 Detection Module That Uses Web Logs

The module that uses web logs for the detection of web bots first extracts the features to be used by the classifiers. To identify these features, we initially studied the related work published in the past seven years (2012–2019) and gathered the most promising ones that have been proposed and that are also applicable in our setting. Table 2 shows the complete list of features, with a short description and relevant studies that use them.

Table 2. Features Extracted from Each Session

Id	Feature	Short description and literature
1	Total requests	Total number of HTTP requests that the agent issued during the session [3, 16, 20, 34, 37, 38, 43].
2	Total session Bytes	The sum of all requested pages' size (in Bytes) in a session [3, 10, 20, 29, 38, 43].
3	HTTP GET requests	Total number of HTTP GET requests issued during the session [4, 6, 10, 20, 34, 43].
4	HTTP POST requests	Total number of HTTP POST requests issued during the session [6, 10, 20, 34, 43].
5	HTTP HEAD requests	Total number of HTTP HEAD requests issued during the session [6, 10, 16, 20, 29, 34, 37, 38, 43].
6	% HTTP 3xx requests	The percentage of HTTP requests that led to an HTTP 3xx code response [4, 10, 20, 43].
7	% HTTP 4xx requests	The percentage of HTTP requests that led to an HTTP 4xx code response [4, 10, 16, 20, 29, 37, 38, 43].
8	% image requests	The percentage of HTTP requests that requested an image. This feature searches for all known image formats' ending [18, 20, 30, 34].
9	% css file request	The percentage of HTTP requests that requested a css file [20, 30].
10	% js requests	The percentage of HTTP requests that requested a JavaScript file [18, 20, 30].
11	HTML-to-image ratio	The number of the requested HTML files divided by the number of requested image files in a session [16, 20, 37, 43].
12	Depth SD	Standard deviation of requested pages' depth (i.e., number of '/' in URL path) [20, 37, 38, 43].
13	Max requests per page	The maximum number of requests to the same page in a session [20].
14	Average requests per page	The average number of requests per page in a session [20].
15	Max number of consecutive sequential HTTP requests	The maximum number of HTTP requested URLs that contain the previously requested URL as a subpart page [20, 43].
16	% of consecutive sequential HTTP requests	The percentage of HTTP requested URLs that contain the previously requested URL as a subpart [16, 20, 37, 38].
17	Session time	The total time (in seconds) between the first and the last HTTP request of the session [3, 4, 20, 29, 34, 43].
18	Browsing speed	The ratio of the total number of requested pages over time (in seconds) [4, 20].
19	SD of inter-request times	Standard deviation of time between successive requests [4, 20].

The next step concerns the selection of the classifiers to be employed in our voting scheme. To this end, four well-established machine learning algorithms, all of whom have been used by other researchers for the web bot detection problem were considered. More specifically, we applied the Support Vector Machine [20, 29, 37], the Random Forest [20, 34], the Adaboost [20, 34], and the Multi-layer Perceptron classifiers [10, 20, 29, 37]. These classifiers construct an ensemble classifier (Voting classifier) that performs a class probability averaging of all the available classifiers [20, 34]. We decided to follow this approach, instead of simply using one of the aforementioned approaches, to provide a more robust detection framework so as to ensure that any shortcomings of an individual classifier will not affect the detection performance. For the implementation of these algorithms the scikit-learn[5] Python library was used.

5.2 Detection Module That Uses Mouse Movements

The module that uses mouse movements is based on CNNs, since they have proven to work very well in identifying patterns, also when employed in relevant research [42].

Inspired by the architecture used in a relevant study [42], as well as considering the nature and complexity of our problem (i.e., detecting line patterns) we chose the architecture presented in Table 3. For the implementation of the Deep Neural Network, the Keras[6] Python library was used.

Table 3. Architecture of the Network for the Detection of Web Bots from Mouse Movements

Layer type	Kernel size / stride	Output Shape	Activation
InputLayer	–	(480, 1320, 1)	–
Conv	3x3 / 2	(239, 659, 64)	ReLU
Conv	3x3 / 2	(119, 329, 64)	ReLU
M-Pool	4x4 / 4	(29, 82, 64)	–
Conv	3x3 / 2	(119, 329, 64)	ReLU
M-Pool	4x4 / 4	(29, 82, 64)	–
Flatten	–	(1920)	–
Dense	–	(2)	Softmax

6 EVALUATION SETUP

To assess the effectiveness of the proposed web bot detection framework that combines both web logs and mouse biometrics, a series of experiments was performed by considering web bots of different sophistication levels. This section describes the evaluation methodology (Section 6.1), the dataset (Section 6.2), the evaluation metrics that we considered (Section 6.3), the configuration of the employed classification algorithms (Section 6.4), and, finally, the configuration of the web bots that we used for the experiments (Section 6.5).

6.1 Evaluation Methodology

The purpose of this work is to examine the effectiveness of the web bot detection framework when faced with malicious web bots such as those considered by our threat model (see Section 3), i.e., web bots that aim to crawl a web server with the purpose of harvesting information of value from that server. For that, we chose a web server that hosted static web pages to generate our dataset. Had we decided on using dynamic and more complex web pages, it would have required a much greater user base to generate a representative dataset. Since there were no requirements regarding the content that the server should host, we decided to use content crawled from Wikipedia.[7]

The evaluation of the framework was performed in two phases. Initially, the framework was evaluated on a testbed web server where the sessions were created by a closed set of participants, i.e., the authors of this work. In the second phase, an expanded version of the web server (including additional content) was visited by 28 additional users (different from the ones used in the first set of experiments). The purpose of the first set of experiments was to evaluate our framework on its ability to detect web bots of different levels of sophistication (i.e., moderate and advanced). The second phase of experiments aimed to evaluate our framework in a more "real-world" scenario, where (i) there is not always a way to isolate suspected web bots of different sophistication levels before they are passed to the detection models, and (ii) the detection framework is usually tested on new users with unseen behaviours. Thus, in the second phase of experiments, we also evaluated our framework in the case where moderate and advanced web bots are merged, and in a user-independent manner, where each user (and thus all their sessions) are used exclusively for either training or testing.

6.1.1 *First Phase of Experiments.* In the first phase of the experiments, we considered two cases for our evaluation: (i) testing the framework on moderate web bots (i.e., web bots that have a browser fingerprint but do not exhibit a humanlike behaviour) and (ii) testing the framework on advanced web bots (i.e., web bots that have a browser fingerprint and exhibit a humanlike behaviour). We do not investigate the case of web bots that do not have a browser fingerprint (i.e., simple bots), because they can trivially be detected using simple rule-based techniques. The purpose of the first phase of experiments was to examine the differences in the framework's performance when web bots present more evasive behaviour.

Furthermore, to gain a better understanding of our detection framework's performance, we evaluated each detection module separately and in combinations. More specifically, besides the two modules and their fusion presented in Section 4.3, we also evaluated the detection modules in a joint OR statement and the detection modules when we simply average their classification probabilities.

Moreover, to account for the fact that in a real-world case scenario we might want to detect web bots as soon as possible (i.e., with a few requests), we examined the performance of our framework over the number of requests as they happen (real time). For the classifier that uses mouse movements, the voting process considers the requests one by one, as they arrive (see Section 4.2.2); thus, no retraining is required. However, for the classifier that uses web logs, we have to retrain the classifiers in each request with all the requests that are available at that time. This is because the classifier that uses web logs considers the whole (available) session before identifying the web bots.

6.1.2 *Second Phase of Experiments.* The second phase of experiments was performed on the same web server, but with further content added. In this second phase of the experiments, the evaluation process aims to reflect an even more realistic scenario. To this end, we initially re-evaluated our framework using the advanced web bots tested before. After that, to account for the fact in a real-world scenario moderate web bots and advanced web bots would be mixed, we evaluated our framework on a dataset that contains both moderate web bots and advanced web bots.

6.2 Dataset

The framework was evaluated on two different versions of the server, the second of which had additional content. For the first phase of the experiments a closed set of participants (i.e., the authors of this work) were used for the generation of the human sessions, whereas in the second phase of the experiments, 28 additional human subjects were used to create the human sessions for the evaluation of the framework. In both cases, the human visitors were presented with web pages containing primarily text and, in some cases, a few images, and they were asked to spend some time on that server, reading part of the content. There were no strict instructions on how to read the content or what content to read to simulate a more real-world case scenario.

6.2.1 *First Phase of the Experiments.* For the first phase of the experiments, the web server that was used to evaluate our framework hosted 61 web pages from five different categories/topics crawled from Wikipedia. The number of pages was heuristically chosen with the objective of providing enough data for the visitors to read from. For the first phase of experiments, 50 human sessions were generated by a closed set of participants, i.e., the authors of this work; in each session we visited the web server for an adequate (not predefined) period of time to generate sufficient data for our experiments. The resulting sessions had between 9 and 20 requests. There were no specific requirements regarding the number of requests. The only goal was to spend sufficient time browsing the server so that an adequate amount of data could be collected. Furthermore, since we evaluated our framework over the number of requests, there was no need for the users to perform a specific number of requests.

The distribution of the total number of requests of the human sessions for the first phase of experiments is presented in Figure 5. Each session was identified based on the PHP session id. Furthermore, we created 50 moderate and 50 advanced web bot sessions that performed a similar number of requests with humans (i.e., a random number between 9 and 20 requests).

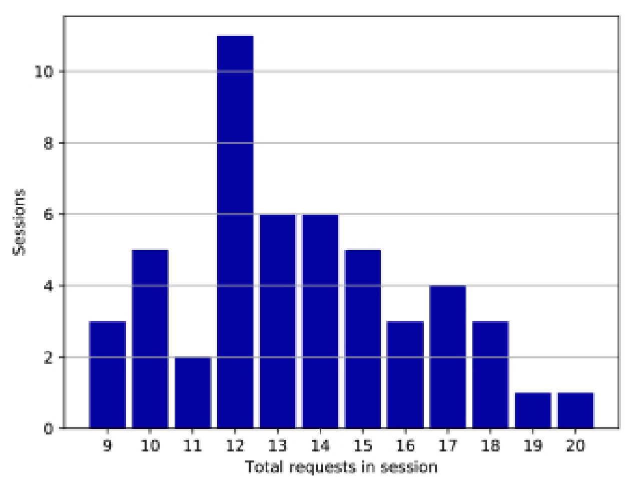


Fig. 5. Distribution of total requests per session for the first phase of experiments.

In the first phase of the experiments, the framework was evaluated on two datasets: (i) the D1, which contains the human sessions and the moderate web bot sessions and (ii) the D2, which contains the human sessions and the advanced web bot sessions. Furthermore, each dataset was split into 70% training and 30% testing for the evaluation of the framework simulating a training and a testing periods. The dataset details are presented in Table 4 in the format of *x*/*y* where *x* is the number of sessions and *y* is the total number of requests during these sessions.

Table 4. Human, Moderate and Advanced Web Bots Sessions and Total Requests for All Sessions of the First Phase of Experiments (Sessions/Requests)

		Bots		D1	D2
	Humans	*Moderate*	*Advanced*	(humans + mod. bots)	(humans + adv. bots)
Train	35 / 456	35 / 431	35 / 527	70 / 887	70 / 983
Test	15 / 172	15 / 196	15 / 239	30 / 368	30 / 411
Total	50 / 628	50 / 627	50 / 766	100 / 1,255	100 / 1,394

6.2.2 *Second Phase of Experiments.* For the second phase of experiments, an expanded version of the same web server was used; this web server hosted a total of 110 web pages from 11 categories/topics (including the content used in the first version of the web server) crawled again from Wikipedia. Over the course of these experiments, 28 additional users were asked to visit our web server, and to create 2 sessions each (thus, the total number of human sessions were 56). We instructed each user to spend about 15–20 minutes per session. All sessions were anonymised and the only information collected was which sessions were generated by the same user.

The distribution of the total number of requests in the sessions of the additional 28 users is presented in Figure 6. Each session was identified based on the PHP session id. However, we also considered a session timeout of 30 minutes; if 30 minutes passed after the last request with a particular session id, any future requests with the same id would be considered as part of a new session.

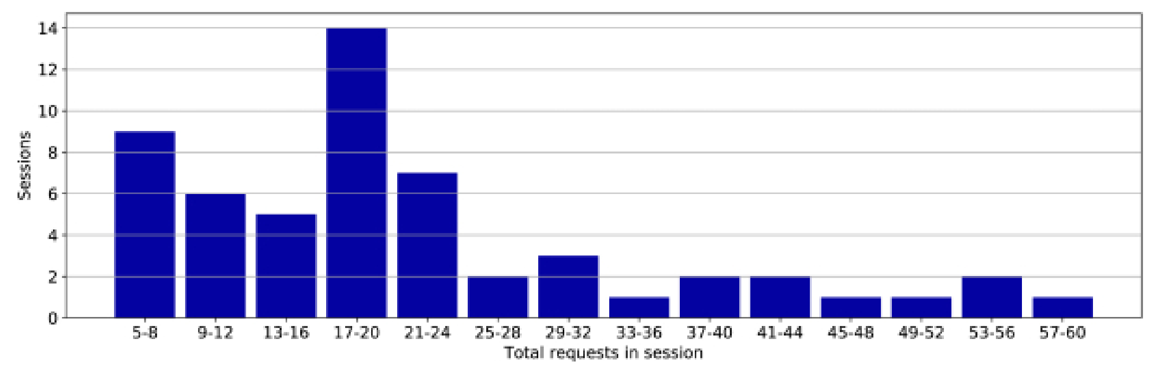


Fig. 6. Distribution of the total requests per session for the second phase of experiments.

In the second phase of the experiments, the framework was tested on two datasets: (i) the D3, which contains the human sessions generated by the additional users and the same number of sessions generated by the same advanced web bots used in the first set of experiments and (ii) the D4, which contains the human sessions generated by the additional users and a mix of moderate and advanced web bot sessions. The rationale behind the selection of the D4 was that in a real-world scenario, such web bots will not usually be isolated based on their evasiveness and thus it is of interest to see how the framework performs under these circumstances. Thus, in D4 we considered 28 moderate and 28 advanced web bots sessions (which is in total equal to the number of human sessions).

Additionally, the second phase of experiments were user-independent (i.e., different users are used for training and for testing) to simulate a more real-world scenario, where the framework will be required to identify new, unseen behaviours as bots or humans. Furthermore, to account for the fact that the selection of which users will be considered for testing may influence the results, we performed a sevenfold cross validation at user level. More specifically, we split the dataset into seven folds, each fold containing the sessions of four users (thus, each fold containing eight human sessions in total, as each human visitor made two sessions) and the same amount of web bot sessions. This enabled our experiments to be user-independent. The final evaluation metrics were calculated as the average of the results of all iterations. Finally, to account for the randomness introduced when the models are trained on GPU (as mentioned in the documentation of the Keras library[8]), the aforementioned process was repeated five times and the average of all runs was considered.

The dataset that was used for the second set of experiments is presented in Table 5.

Table 5. Human, Moderate and Advanced Web Bots Sessions and Total Requests for All Sessions of the Second Phase of Experiments

	D3			D4	
	Humans	**adv. bots**	**humans+adv.**	**mod.+adv.**	**humans+mod.+adv.**
sessions	56	56	112	56	112
requests	1211	754	1965	734	1945

6.3 Evaluation Metrics

To assess the effectiveness of the proposed web bot detection framework, we calculated the accuracy, a common metric used in the web bot detection problem [29, 30, 37, 41]. Furthermore, to gain a better understanding of our framework's performance, we also calculated the precision, recall, and F-score (the harmonic mean of precision and recall) evaluation metrics that are also commonly used for the evaluation of web bot detection frameworks [3, 16, 18, 20, 29, 34, 37].

6.4 Classification

The framework combines two classification modules for the detection of web bots: (i) one that uses web logs and (ii) one that uses mouse movements; the parameters of these classification modules are presented below.

6.4.1 *Classifiers Using Web Logs.* This module employs an ensemble of four well-established machine learning algorithms: Support Vector Machine, Random Forest, Adaboost, and the Multi-layer Perceptron.

For the selection of the parameters of these four classifiers, we performed an exhaustive search over specified parameter values (grid search) and chose the ones with the highest accuracy using a twofold cross validation on the training data [20]. Furthermore, the data (both training and testing) are scaled when inserted to the classification algorithms to avoid the domination of some features over the others. For the scaling process, initially we calculated the mean and the standard deviation of the training data for each feature in the training set. These values were used to scale the data of each feature by subtracting the calculated mean value and then dividing by the standard deviation [20]. The final values of the parameters are presented in Table 6.

Table 6. The Parameters of Classification Algorithms That Use Web Logs

Classification Algorithm	**D1: Humans – Moderate bots**	**D2: Humans – Advanced bots**
SVC	RBF kernel, C=1, gamma=0.03125, tol=0.001	RBF kernel, C=16, gamma=0.002, tol=0.001
MLP Classifier	ReLU activation, SGD solver, a=0.001, b1=0.1, b2=0.1, e=1e-08, hidden layer sizes: (100, 50), constant learning rate	ReLU activation, adam solver, a=0.001, b1=0.9, b2=0.9, e=1e-08, hidden layer sizes: (100, 50), constant learning rate
Random Forest	estimators=200, Gini criterion, max features $= \sqrt{\#features}$, min samples per leaf = 1, min samples split = 2, max depth=10, out-of-bag samples used	estimators = 200, Gini criterion, maxfeatures= $\sqrt{\#features}$, min samples per leaf = 4, min samples split = 10, max depth=10, out-of-bag samples used
Adaboost	Decision Tree Classifier as base estimator, estimators=1250, decision entropy criterion, no max depth, max features= $\sqrt{\#features}$, “best” split strategy, learning rate=0.5	Decision Tree Classifier as base estimator, estimators=1250, decision gini criterion, no max depth, max features= $\sqrt{\#features}$, “best” split strategy, learning rate=1

To select the best performing features for our voting (ensemble) classifier, we used the Sequential Feature Selection technique, a greedy algorithm that can be used to reduce the feature space [20, 28]. More specifically, we used the **Sequential Forward Floating Selection (SFFS)** technique [28], in which we start with no features and then add the most effective features by testing them one by one on the training data. Furthermore, in each iteration there is an extra exclusion step where features might be removed so that a larger number of feature subset combinations can be sampled.

The accuracy of the SFFS for both the D1 and D2 datasets is presented in Figure 7. Based on this, we selected the set of features that resulted in the highest accuracy in the training set for both the D1 and D2. For both the D1 and D2 the number of features that present the highest accuracy are 14. These selected feature indexes are {0, 1, 3, 4, 6, 8, 10, 11, 12, 13, 14, 15, 17, 18} for D1 and {0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 14, 15, 16, 18} for D2 (see Table 2 for the corresponding features).

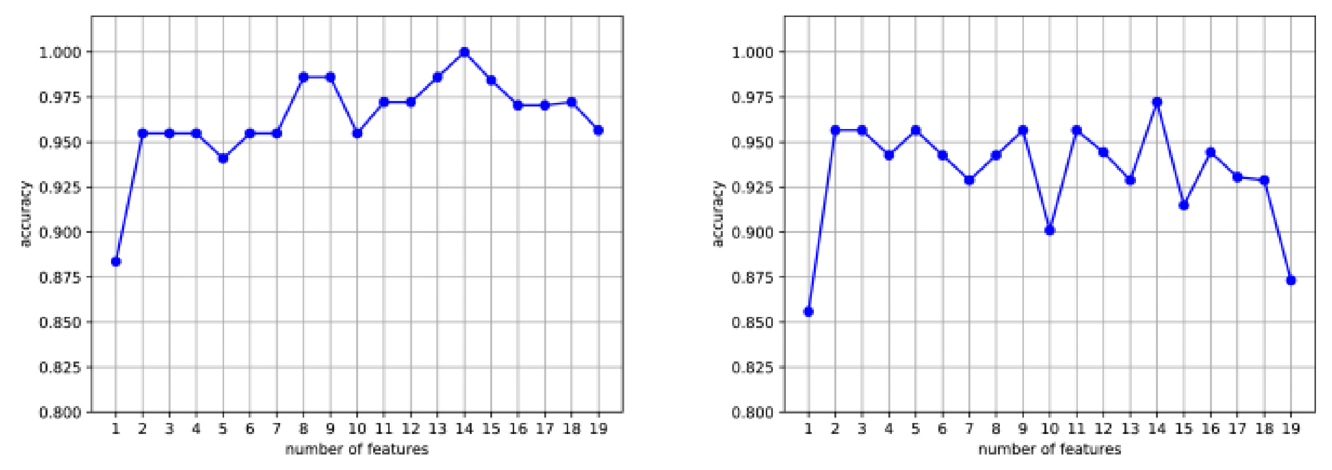


Fig. 7. Sequential Forward Floating Selection for D1 (left) and D2 (right).

Preliminary experiments in our datasets showed that the aforementioned classifiers (both the individual ones and the ensemble one) were often able to achieve similar results. However, there were cases in all datasets (i.e., specific numbers of requests) where the performance of an individual classifier was worse than the combination using the ensemble (voting method); the precise number of requests for which we observed this performance varies for each classifier. This indicates the need to use the ensemble classifier presented above to further increase the robustness of our approach.

6.4.2 *Classification Using Mouse Movements.* This module utilises a Deep Neural Network architecture that utilises CNN layers for the detection of web bots based on their mouse trajectories. After a manual inspection or our datasets, and to account for the high memory usage of the input matrices (1,080 × 1,920), we took advantage of the fact that human visitors did not perform mouse movements on the edges of the web pages by performing a center cropping with 300 pixel offset.

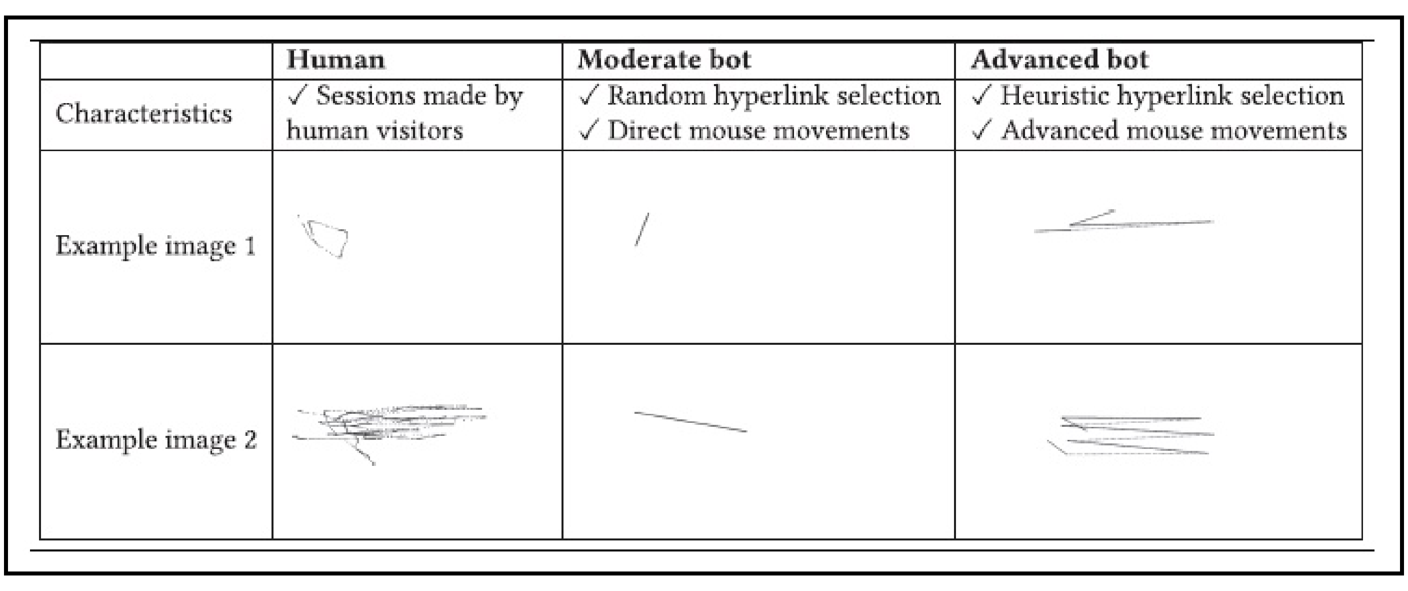
6.4.3 *Fusion.* Each detection module produces a score between 0 and 1. A high score means that a visitor is very likely to be a bot, and a low score that the visitor is very likely to be a human. The framework combines these scores as shown in Equation (1). The values of the thresholds for which only the mouse movement detection module was used were initially selected heuristically and then fine-tuned based on the accuracy of the framework on the training data of D2. The final values that were selected are $thres_h$=0.7 and $thres_l$=0.3. Furthermore, when the two detection modules are combined, we take the average of the detection scores, so $w_{mv} = w_{wl} = 0.5$. This was not fine-tuned, and further optimisations could be performed. If the total score is greater than or equal to 0.5, then we consider the session to be a bot. Otherwise, we consider the session to be human. These values were selected heuristically.

6.5 Web Bots and Their Configuration

The purpose of this research is to examine the effectiveness of the framework in detecting malicious web bots of different sophistication levels. Malicious web bots can vary from simple scripts to bots that automate browsers and present a humanlike behaviour [14]. Thus, in this research we evaluated our framework based on its effectiveness in detecting web bots of two levels of sophistication: (i) the moderate web bots that have a browserlike fingerprint but exhibit no humanlike behaviour and (ii) the advanced web bots that present both a browser fingerprint and a humanlike behaviour. We did not evaluate the performance of our framework on simple web bots (i.e., simple scripts), since they perform no mouse movements and thus they will always be detected.

As there is neither a universal definition of how the moderate and advanced web bots behave, nor are there any existing tools that we can use to generate them, we based these behaviours on the information available in literature [11, 21, 27], as well as in reports of web bot detection companies [15]. More specifically, we consider the moderate web bots to have no intelligence, meaning that they follow hyperlinks randomly and by performing direct mouse movements between those links. Advanced web bots, however, exhibit some intelligence by performing a heuristic hyperlink selection, as well as some more advanced mouse movements based on the web pages' content. Table 7 summarises the behaviour of humans, moderate web bots, and advanced web bots, by visualising two example mouse movement matrices for each type of visitor. The details of how these behaviours are generated are presented in Section 6.5.1 for the moderate web bots and Section 6.5.2 for the advanced web bots.

Table 7. Browsing Behaviour of Human, Moderate, and Advanced Web Bots



	Human	Moderate bot	Advanced bot
Characteristics	✓ Sessions made by human visitors	✓ Random hyperlink selection ✓ Direct mouse movements	✓ Heuristic hyperlink selection ✓ Advanced mouse movements
Example image 1			
Example image 2			

6.5.1 *Moderate Web Bots.* The moderate web bots were programmed to follow the same number of web pages as the humans visited, which is a random number between 9 and 20 (see Section 6.2.1). They follow hyperlinks by extracting all the available hyperlinks from each web page and randomly (with equal probability per page) choosing one. Additionally, moderate web bots present mouse movements that directly connect their current position with the position of the next hyperlink that they will follow. Furthermore, the mouse move "step" (i.e., the distance between each point/pixel that the mouse, when controlled by the web bot, hovers over) is 1, resulting in a continuous straight line, unlike advanced web bots and human users.

6.5.2 *Advanced Web Bots.* The advanced web bots, similarly to moderate web bots, were also programmed to follow the same number of web pages as the authors visited, which is a random number between 9 and 20 (see Section 6.2.1). However, instead of randomly selecting the hyperlinks to follow like moderate bots, advanced web bots are more likely to follow hyperlinks from within the same topic. This is also common to human users when visiting websites such as Wikipedia. Additionally, advanced web bots can emulate "reading" part of a web page by performing mouse movements in a left to right direction and back, as if to follow the direction of the text (like humans sometimes do). As a result, the time between requests is also adjusted based on whether they are "reading" the web page or not, and based on the content of the web page that is read [11].

To achieve the aforementioned behaviours, a number of heuristically selected parameters were used. Next, we present these specific configurations and parameters regarding the hyperlink selection process and the performed mouse movements.

For the selection of which hyperlink to follow, advanced web bots initially select a random hyperlink from all the available hyperlinks in each web page. Since the structure of the web server is similar to the one of Wikipedia, the majority of the hyperlinks in a web page are often on the same semantic topic as their parent page (i.e., the page including those hyperlinks). Thus, even in a random hyperlink selection policy (like the one followed by the moderate web bots), bots have a tendency of remaining on the same semantic topic. To further increase the probability of staying on the same topic, when advanced bots try to visit a new topic they have a 50% probability of being forced to choose again (i.e., repeat the hyperlink selection process) instead of visiting the new topic.

Additionally, advanced web bots have an 80% probability of simulating a "reading" of the web page (i.e., performing mouse movements hovering over the text as if the bots are reading). The lines to be read are calculated based on the text size using the following equation:

$$lines\_to\_read = \frac{content\_length - template\_length}{length\_to\_lines}, \tag{2}$$

where the $content\_length$ is the length of the web page when considered as a string variable, the $template\_length$ represents the length of the text that belongs to the part of the web page that remains constant for all requests (i.e., the web page theme or template), and the $length\_to\_lines$ is a weighting factor that allows the transformation of content length to lines based on the content size.

To achieve a mouse movement behaviour as the one presented in Table 7 for advanced web bots, we heuristically selected a set of parameters. This behaviour depends on three factors: (i) how many lines the web bot will "read" (i.e., hover across the page from the left side to the right side in an approximately horizontal line, like human users do), (ii) given a starting point, which will be the ending point of the line that represents the "reading" function, and (iii) which will be the next point that the bot will go to after finishing "reading" a line, i.e., the next line's starting point. The first is calculated using Equation (2). We selected $content\_length = 1500$ and $length\_to\_lines = 500$ based on the structure of the web pages. For the second requirement, instead of allowing the bot to hover over the line until its end, we deduct a random number between (0,200) from the horizontal axis coordinate of the ending point to simulate the human trait of skipping the end of a line. We also add a random number between (50,100) to the vertical axis coordinate of the ending point, because humans do not move the mouse in an exact straight line. After finishing a line, the web bot uses the starting point of the newly finished line to calculate the coordinates of the next starting point. The next starting point will be the old starting point with its coordinates incremented by a random number between (0,50) for the abscissa (x-axis) and a random number between (50,100) for the ordinate (y-axis) respectively.

Finally, the advanced web bots perform a "step" of 8 when going from the left to the right of each line (simulating a "normal reading") and a "step" of 18 when going back to the start of the line (simulating a mouse move without reading). All the aforementioned parameters were chosen heuristically with the purpose of presenting a more humanlike mouse movement behaviour.

6.5.3 *Software for Generating the Web Bots.* For the purposes of this work, we generated the web bots using the Selenium[9] browser automation software in its default configuration to present an approximate browser fingerprint, and to enable the creation of a humanlike browsing behaviour (i.e., the generation of clicks, and mouse movements). However, in a real-world scenario, such web bots must be further configured [5] or regular browsers could be utilised [2] to avoid deterministic detection based on visitors' fingerprints. For the purposes of this research, we assume that the fingerprint generated by Selenium is indistinguishable from a browser fingerprint.

7 RESULTS

In this section we present the results of the evaluation of our framework. The framework was tested in two phases: the first phase, where we evaluate the framework in its ability to detect advanced web bots as opposed to moderate ones (Section 7.1), and the second phase, where the framework is evaluated in a more real-world scenario, where suspected moderate and advanced web bots cannot be always isolated before being passed to the detection models (Section 7.2).

Regarding the first set of experiments, we initially present the overall performance of our framework in the dataset generated by the authors of this work when the modules are used alone or in combinations (Section 7.1.1). After that, and since we want the servers to identify web bots with as few requests as possible (i.e., online, before the session ends), we also examined the effectiveness of the aforementioned classification models per request, i.e., initially considering only the first request in each session and gradually increasing the number of requests considered (Section 7.1.2). Finally, in the second set of experiments, we evaluated our framework over time on the sessions generated by the additional users (Section 7.2).

7.1 First Phase of Experiments

In the first set, we performed a series of experiments to evaluate the performance of our framework in total and over the number of requests. The datasets that were utilised for that were the D1 and D2 (see Section 6.2.1). The results of our framework are presented below.

7.1.1 *Overall Performance.* To evaluate the overall performance of our framework, we calculated the accuracy as well as the precision, recall, and F-score for both the web bot and the human class. Furthermore, to better illustrate the performance of our framework, we considered several combinations of our detection modules. More specifically, we considered (i) using only the module that uses web logs, (ii) using the module that uses only mouse movements, (iii) combining the two detection modules in an OR statement (i.e., a visitor is a bot when at least one module classifies them as a bot), (iv) averaging the classification probability of the detection modules, and (v) fusing them based on the Equation (1). The results are presented in Table 8.

Table 8. Evaluation of the Web Bot Detection Framework per Session for D1 and D2

		web logs	mouse movements	web logs OR mouse movements	average	fusion
		D1 / D2	D1 / D2	D1 / D2	D1 / D2	D1 / D2
	Precision	0.93 / 0.92	1.00 / 0.88	0.94 / 0.83	1.00 / 1.00	1.00 / 1.00
bot class	*Recall*	0.87 / 0.80	1.00 / 1.00	1.00 / 1.00	1.00 / 0.93	1.00 / 1.00

		web logs	mouse movements	web logs OR mouse movements	average	fusion
		D1 / D2	D1 / D2	D1 / D2	D1 / D2	D1 / D2
	F-score	0.90 / 0.86	1.00 / 0.94	0.97 / 0.91	1.00 / 0.97	1.00 / 1.00
	Precision	0.88 / 0.82	1.00 / 1.00	1.00 / 1.00	1.00 / 0.94	1.00 / 1.00
human class	*Recall*	0.93 / 0.93	1.00 / 0.87	0.93 / 0.80	1.00 / 1.00	1.00 / 1.00
	F-score	0.90 / 0.88	1.00 / 0.93	0.97 / 0.89	1.00 / 0.97	1.00 / 1.00
	Accuracy	0.90 / 0.87	1.00 / 0.93	0.97 / 0.90	1.00 / 0.97	1.00 / 1.00

As expected, detecting advanced web bots is more difficult than detecting moderate web bots for all detection modules, since advanced web bots try to present a more humanlike behaviour. Furthermore, the module that uses mouse movements achieves higher accuracy and F-score than the one that uses web logs. In the OR combination, if a module incorrectly classifies a user as a bot, the error propagates to the result. For this reason, along with the fact that the web log module performs worse than the mouse movement module, the OR combination of the modules yields lower effectiveness than the mouse movement module. However, averaging their classification probabilities hides these errors, because the correct detection module in each case exhibits either a very high or a very low classification probability while the incorrect one does not. Finally, when fusing their results as presented in Equation (1), by using solely the mouse movement detection module when its classification probability is either very high or very low, the framework classifies all visitors correctly in our test dataset for both the moderate and the advanced web bots.

7.1.2 *Performance over the Number of Requests*. Next, we calculated the accuracy of the framework over the number of requests. We used the same module combinations as in Section 7.1.1 and performed an iterative process where we initially considered only the first request in each session and gradually increased the number of requests considered. When a session reached the maximum number of requests available, we stopped increasing the number of requests considered for that session. The results are presented in Figure 8.

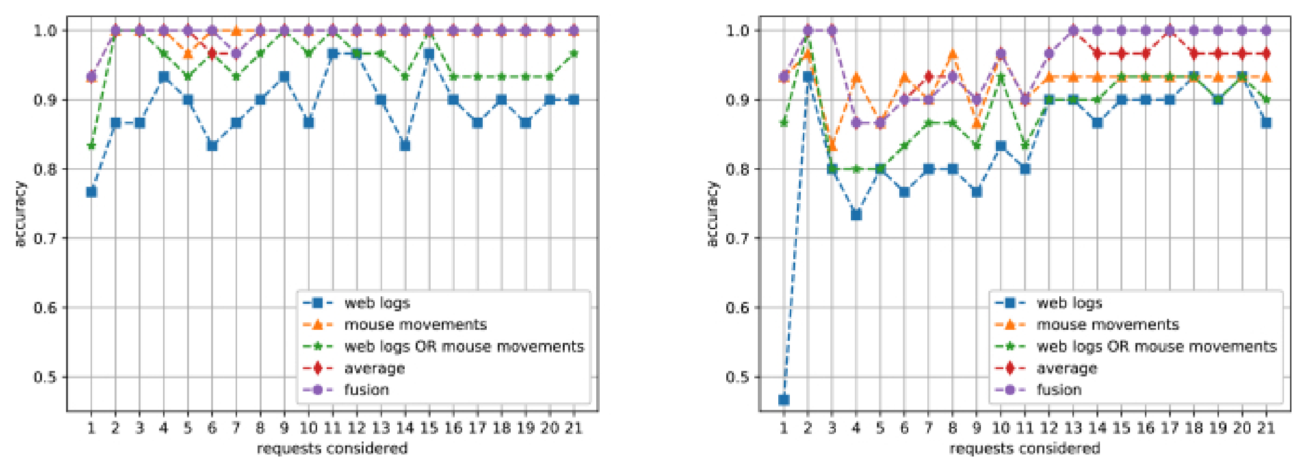


Fig. 8. Accuracy over requests for D1 (left) and D2 (right).

Again, the framework performs better when faced with moderate web bots as opposed to advanced web bots for all the detection modules. Furthermore, detection using mouse movements is more effective compared to the one that uses web logs. For D1, the mouse movement module can be used as a stand-alone module, as it achieves 100% accuracy from the first requests. The same behaviour is also achieved when averaging the classification probabilities of the detection module or fusing them based on the Equation (1). In contrast, the web log module alone or in combination with the mouse movement detection module in an OR statement (in which errors from the web log module are propagated to its output) are less effective approaches. In D2, both the detection module that uses web logs and the one that uses mouse movements present a lower effectiveness in comparison with D1. As a result, the performance of the combination of those modules decreases. However, the fusion of those modules retains its effectiveness, which indicates that the two different detection approaches complement each other even in the cases where one of the individual classifiers fails.

Moreover, an unexpected outcome was that in both D1 and D2, the accuracy score did not increase as the number of considered requests grew larger. More specifically, in D2, the accuracy of all detection modules is very high when using only two requests. This indicates that, while advanced web bots may present a long-term humanlike behaviour, when tested on a few requests, their behaviour varies from the norm, which makes them easier to detect.

Finally, we tested the statistical significance of the difference in the performance of each detection module and their combinations using a paired, one tail, *t*-test with $a = 0.01$. For the D1, the performance difference is statistically significant in all cases except from the "mouse movements," "average," and "fusion," which, as shown in Figure 8, perform very similarly. For the D2, the performance differences in all cases are statistically significant.

7.2 Second Phase of Experiments

In the second phase of the experiments, the final accuracy was calculated as the average accuracy across all iterations (see Section 6.2.2). The results are presented in Figure 9, where the accuracy is plotted per request.

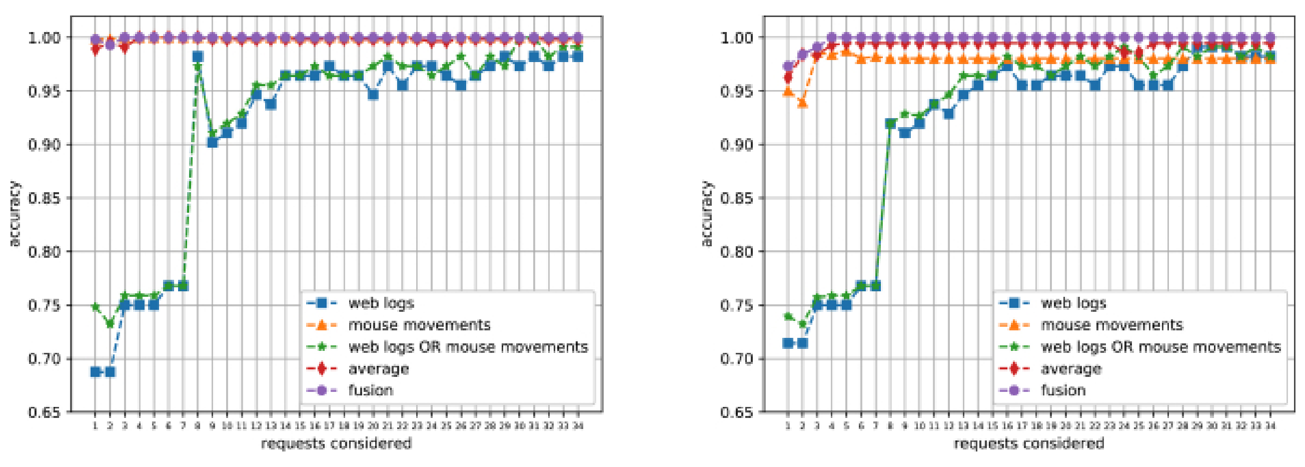


Fig. 9. Accuracy over requests for D3 (left) and D4 (right).

The results in the second phase of experiments indicate that both the detection module that uses web logs and also the detection module that uses mouse movements performed considerably better, with the latter achieving a very high performance even from the very first requests compared to the ones of the first phase of experiments on the same number of requests. Additionally, we observed that the performance of the detection modules and their combinations stabilises after around 30 requests. Through a manual investigation of the two datasets used in the two phases of the experiments, we observe that the new users exhibited a larger variety of behaviours. This benefited our framework and made it easier to detect advanced web bots, as they are more similar to each other than they are to human users.

The aforementioned observation is consistent with the behaviour of the framework when mixing moderate with advanced web bots in D4. Given that D4 considers both moderate and advanced web bots, there is an increased heterogeneity in the possible behaviours of web bots and this affects negatively the performance of the individual detection modules. However, while the performance of the individual modules is affected by mixing the two types of bots, the fusion of the detection modules achieves a high accuracy.

Moreover, the combination of the two detection modules in an OR statement performs slightly better than the web log detection module (which indicated that the errors from the web log detection module propagate to its output), while the average and the fusion of the two detection modules perform slightly better than the mouse movements detection module.

Finally, we tested the statistical significance of the difference in the performance of each detection module and their combinations using a paired, one tail, *t*-test with $a = 0.01$. For the D3, the performance difference is statistically significant in all cases except from the "mouse movements," "average," and "fusion." For the D4, the performance differences in all cases are statistically significant.

8 DISCUSSION

The ever increasing use of web bots for malicious purposes has led to a need for new and more sophisticated methods for web bot detection. As websites often implement some form of web bot detection, attackers have responded by creating advanced web bots that can evade detection on these websites. Even though current web bot detection techniques proposed in literature are highly accurate for simple and moderate web bots, they have not been thoroughly tested on advanced web bots.

This study proposed a web bot detection framework that can accurately detect advanced web bots. It is an amalgamation of two detection modules, one that extracts information from web logs to determine if a visitor is a human or a web bot and one that detects web bots based on their mouse movements. The framework combines the results of each module in a novel way to capture the different temporal characteristics of the web logs and the mouse movements, as well as the spatial characteristics of the mouse movements.

When used individually, the detection module that uses mouse movements is more effective than the web log detection module, because human mouse movements are more difficult to simulate in comparison with human browsing behaviour (regarding the web pages visited). Furthermore, while some web bots were able to bypass at least one module, none of the web bots we tested were able to bypass both of them. This means that it is more difficult for web bots to present humanlike mouse movements and browsing behaviour simultaneously. However, simply classifying any session as a bot if either of the detection modules identifies it as a bot will not be sufficient, since, based on our results, it can lead to the misclassification of human sessions.

Additionally, this research examines web bots of specific evasiveness levels. We show that the framework was able to detect advanced web bots more effectively when a more broad dataset of users with different behaviours was used. This indicates that advanced web bots should be modelled to present a broader set of behaviours (i.e., simulating different types of users). However, the creation of such web bots is challenging, since the more advanced a web bot is, the more complicated behaviour must be generated. Thus, this work raises the question of how a set of evasive behaviours for advanced web bots can be generated effortlessly.

Moreover, the current version of the framework can detect web bots only if they allow the specific JavaScript file that tracks users mouse movements to run. Alternative approaches can be utilised to the collection of mouse movements that do not use JavaScript. However, in both cases, a (malicious) actor could block such tracking techniques. We believe that, depending on the application of the framework, such actors could be categorised as potentially malicious and the web server could apply additional bot detection techniques to them (e.g., techniques that require human interaction [15]). Furthermore, our framework has not been designed to protect against replay attacks, in which human behaviour may be recorded and then repeated by a web bot. These attacks are more time consuming, and they are specific to each web server and its web pages' structure.

Finally, when applying the proposed web bot detection framework to a web server with numerous visitors, its effect in the server's efficiency needs to be considered. The collection of all the visitors' mouse movements is a resource demanding process. Thus, we believe that the aforementioned approaches should only be performed in the first few requests of the visitors and not during the entire session. Our framework achieves high accuracy with a small number of requests, which makes it suitable for online detection.

9 CONCLUSIONS AND FUTURE WORK

This work proposed a framework for detecting web bots that present a browser fingerprint and a humanlike behaviour. The detection framework combines two detection modules, (i) one that uses web logs and (ii) one that uses mouse movements. It initially calculates a score using only the mouse movement detection module, and, if the score is below a predefined threshold (i.e., if the result is uncertain), it calculates the average of the scores from the two detection modules.

We evaluated our framework on a test web server that was accessed by human visitors and simulated malicious web bots. The bot traffic originated from web bots of different sophistication levels: (i) *moderate web bots* that have a browser fingerprint (which includes performing mouse movements) and (ii) *advanced web bots* that both have a browser fingerprint and also exhibit a humanlike behaviour regarding the web pages visited and the mouse movements performed.

The results have shown that the detection module that uses mouse movements is more effective in general than the one that uses web logs, and the fusion of the two detection modules when prioritising the module that uses mouse movements is more resistant to errors compared with other simple combination approaches. This indicates that simulating humanlike mouse movements is more difficult than simulating humanlike browsing behaviour and simulating both simultaneously requires a higher complexity than the ones tested in this work. Furthermore, we have shown that the proposed approach is suitable for the online detection of web bots, as it achieves high effectiveness with very few requests.

Future work includes the evaluation of the proposed framework on more sophisticated advanced web bots. The currently proposed advanced web bots can be further improved to evade detection. However, based on our results, it is difficult to heuristically define such an evasive behaviour. To this end, we are planning to examine the performance of our framework against advanced web bots that actively adjust their behaviour to avoid detection based on whether they have been detected or not. In this case, the framework will also have to adjust its detection models to address the changes in the bots' behaviour. Additionally, to further increase the performance of the framework against more advanced web bots, novel techniques for the automatic adaptation of the weights and thresholds of the fusion process will be examined [25]. Furthermore, the framework needs to be extended so that it can detect web bots that perform replay attacks (i.e., record and repeat the browsing behaviour of a human visitor) by examining similarities in visitor behaviours.

REFERENCES

[1] Akamai. 2018. Akamai's Bot Manager—Advanced Strategies to Flexibly Manage the Long-term Business and IT Impact of Bots. Retrieved from **https://www.akamai.com/us/en/multimedia/documents/product-brief/bot-manager-product-brief.pdf (https://www.akamai.com/us/en/multimedia/documents/product-brief/bot-manager-product-brief.pdf)**. Navigate to

[2] Ismail Akrout, Amal Feriani, and Mohamed Akrout. 2019. Hacking Google recaptcha v3 Using Reinforcement Learning. arXiv:1903.01003. Retrieved from **https://arxiv.org/abs/1903.01003 (https://arxiv.org/abs/1903.01003)**. Navigate to

[3] Shafiq Alam, Gillian Dobbie, Yun Sing Koh, and Patricia Riddle. 2014. Web bots detection using particle swarm optimization based clustering. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'14)*. IEEE, 2955–2962. Navigate to

[4] Yasmin A. AlNoamany, Michele C. Weigle, and Michael L. Nelson. 2013. Access patterns for robots and humans in web archives. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, 339–348. Navigate to

[5] Babak Amin Azad, Oleksii Starov, Pierre Laperdrix, and Nick Nikiforakis. 2020. Web Runner 2049: Evaluating Third-Party Anti-bot Services. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 135--159. DOI: **https://doi.org/10.1007/978-3-030-52683-2_7 (https://doi.org/10.1007/978-3-030-52683-2_7)** Navigate to

[6] Quan Bai, Gang Xiong, Yong Zhao, and Longtao He. 2014. Analysis and detection of bogus behavior in web crawler measurement. In *Proceedings of the Second International Conference on Information Technology and Quantitative Management (ITQM'14)*. Elsevier, 1084--1091. DOI: **https://doi.org/10.1016/j.procs.2014.05.363 (https://doi.org/10.1016/j.procs.2014.05.363)** Navigate to

[7] Anshul Bhargav and Munish Bhargav. 2014. Pattern discovery and users classification through web usage mining. In *Proceedings of the International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT'14)*. IEEE, 632–636. Navigate to

[8] David Bianco. 2013. The pyramid of pain. *Enterprise Detection & Response* (2013). **http://detect-respond.blogspot.com/2013/03/thepyramid-of-pain.html (http://detect-respond.blogspot.com/2013/03/thepyramid-of-pain.html)**. Navigate to

[9] Kevin Bock, Daven Patel, George Hughey, and Dave Levin. 2017. unCaptcha: A low-resource defeat of recaptcha's audio challenge. In *Proceedings of the 11th {USENIX} Workshop on Offensive Technologies ({WOOT}'17)*. Navigate to

[10] Alberto Cabri, Grażyna Suchacka, Stefano Rovetta, and Francesco Masulli. 2018. Online web bot detection using a sequential classification approach. In *Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS'18)*. IEEE, 1536–1540. Navigate to

[11] Michele Campobasso, Pavlo Burda, and Luca Allodi. 2019. CARONTE: Crawling adversarial resources over non-trusted, high-profile environments. In *Proceedings of the 2019 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW'19)*. IEEE, 433–442. Navigate to

[12] Zi Chu, Steven Gianvecchio, and Haining Wang. 2018. Bot or human? A behavior-based online bot detection system. In *From Database to Cyber Security: Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday*. 432–449. DOI: **https://doi.org/10.1007/978-3-030-04834-1_21 (https://doi.org/10.1007/978-3-030-04834-1_21)** Navigate to

[13] Zibusiso Dewa and Leandros A Maglaras. 2016. Data mining and intrusion detection systems. *vol* 7 (2016), 62–71. Navigate to

[14] Distil Networks. 2018. 2018 BAD BOT REPORT: The Year Bad Bots Went Mainstream. Retrieved from **https://resources.distilnetworks.com/white-paper-reports/2018-bad-bot-report (https://resources.distilnetworks.com/white-paper-reports/2018-bad-bot-report.)**. Navigate to

[15] Distil Networks. 2019. 2019 BAD BOT REPORT: The Bot Arms Race Continues. Retrieved from **https://resources.distilnetworks.com/white-paper-reports/bad-bot-report-2019 (https://resources.distilnetworks.com/white-paper-reports/bad-bot-report-2019.)**. Navigate to

[16] Wang Dong, Xi Lei, Zhang Hui, Liu Hebing, Zhang Hao, and Song Ting. 2015. Web robot detection with semi-supervised learning method. In *3rd International Conference on Material, Mechanical and Manufacturing Engineering (IC3ME'15)*. Atlantis Press, 2123--2128. Navigate to

[17] Derek Doran and Swapna S. Gokhale. 2012. A classification framework for web robots. *J. Assoc. Inf. Sci. Technol.* 63, 12 (2012), 2549–2554. Navigate to

[18] Derek Doran and Swapna S. Gokhale. 2016. An integrated method for real time and offline web robot detection. *Expert Syst.* 33, 6 (2016), 592–606. Navigate to

[19] Javad Hamidzadeh, Mahdieh Zabihimayvan, and Reza Sadeghi. 2017. Detection of Web site visitors based on fuzzy rough sets. *Soft Comput.* 22, 7 (2018), 2175--2188. DOI: **https://doi.org/10.1007/s00500-016-2476-4 (https://doi.org/10.1007/s00500-016-2476-4)** Navigate to

[20] Christos Iliou, Theodoros Kostoulas, Theodora Tsikrika, Vasilis Katos, Stefanos Vrochidis, and Yiannis Kompatsiaris. 2019. Towards a framework for detecting advanced Web bots. In *Proceedings of the 14th International Conference on Availability, Reliability and Security (ARES'19)*.18:1–18:10. DOI: **https://doi.org/10.1145/3339252.3339267 (https://doi.org/10.1145/3339252.3339267)** Navigate to

[21] Christos Iliou, Theodora Tsikrika, Stefanos Vrochidis, and Yiannis Kompatsiaris. 2017. Evasive focused crawling by exploiting human browsing behaviour: A study on terrorism-related content. In *Proceedings of the 1st International Workshop on Cyber Deviance Detection co-located with the 10th International Conference on Web Search and Data (Mining CyberDD @ WSDM 2017)*. Navigate to

[22] Hugo Jonker, Benjamin Krumnow, and Gabry Vlot. 2019. Fingerprint surface-based detection of web bot detectors. In *Proceedings of the European Symposium on Research in Computer Security*. Springer, 586–605. Navigate to

[23] Pierre Laperdrix, Nataliia Bielova, Benoit Baudry, and Gildas Avoine. 2020. Browser fingerprinting: A survey. *ACM Trans. Web* 14, 2 (2020), 1–33. Navigate to

[24] Pierre Laperdrix, Walter Rudametkin, and Benoit Baudry. 2016. Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints. In *Proceedings of the 2016 IEEE Symposium on Security and Privacy (SP'16)*. IEEE, 878–894. Navigate to

[25] Borui Li, Wei Wang, Yang Gao, Vir V. Phoha, and Zhanpeng Jin. 2020. Wrist in motion: A seamless context-aware continuous authentication framework using your clickings and typings. *IEEE Trans. Biometr. Behav. Identity Sci.* 2, 3 (2020), 294--307. DOI: **https://doi.org/10.1109/TBIOM.2020.2997004 (https://doi.org/10.1109/TBIOM.2020.2997004)** Navigate to

[26] G. Neelima and Sireesha Rodda. 2016. Predicting user behavior through sessions using the web log mining. In *Proceedings of the International Conference on Advances in Human Machine Interaction (HMI'16)*. IEEE, 1–5. Navigate to

[27] Sergio Pastrana, Daniel R. Thomas, Alice Hutchings, and Richard Clayton. 2018. CrimeBB: Enabling cybercrime research on underground forums at scale. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web (WWW'18)*. 1845–1854. DOI: **https://doi.org/10.1145/3178876.3186178 (https://doi.org/10.1145/3178876.3186178)** Navigate to

[28] Pavel Pudil, Jana Novovicová, and Josef Kittler. 1994. Floating search methods in feature selection. *Pattern Recogn. Lett.* 15, 10 (1994), 1119–1125. DOI: **https://doi.org/10.1016/0167-8655(94)90127-9 (https://doi.org/10.1016/0167-8655(94)90127-9)** Navigate to

[29] Stefano Rovetta, Alberto Cabri, Francesco Masulli, and Grażyna Suchacka. 2017. Bot or not? A case study on bot recognition from web session logs. In *Proceedings of the Italian Workshop on Neural Nets*. Springer, 197–206. Navigate to

[30] H Nathan Rude and Derek Doran. 2015. Request type prediction for web robot and internet of things traffic. In *Proceedings of the IEEE 14th International Conference on Machine Learning and Applications (ICMLA'15)*. IEEE, 995–1000. Navigate to

[31] Michael Schwarz, Florian Lackner, and Daniel Gruss. 2019. JavaScript template attacks: Automatically inferring host information for targeted exploits. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'19)*. Navigate to

[32] Merve Baş Seyyar, Ferhat Özgür Çatak, and Ensar Gül. 2018. Detection of attack-targeted scans from the Apache HTTP server access logs. *Appl. Comput. Inf.* 14, 1 (2018), 28--36. Navigate to

[33] Dilip Singh Sisodia and Shrish Verma. 2012. Web usage pattern analysis through web logs: A review. In *Proceedings of the International Joint Conference on Computer Science and Software Engineering (JCSSE'12)*. IEEE, 49–53. Navigate to

[34] Dilip Singh Sisodia, Shrish Verma, and Om Prakash Vyas. 2015. Agglomerative approach for identification and elimination of web robots from web server logs to extract knowledge about actual visitors. *J. Data Anal. Inf. Process.* 3, 01 (2015), 1. Navigate to

[35] Suphannee Sivakorn, Iasonas Polakis, and Angelos D. Keromytis. 2016. I am robot: (Deep) learning to break semantic image CAPTCHAs. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P'16)*. 388–403. DOI: **https://doi.org/10.1109/EuroSP.2016.37 (https://doi.org/10.1109/EuroSP.2016.37)** Navigate to

[36] Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. 2016. I'm not a human: Breaking the Google reCAPTCHA. In *Black Hat ASIA 2016*. 1--12. Navigate to

[37] Dusan Stevanovic, Aijun An, and Natalija Vlajic. 2012. Feature evaluation for web crawler detection with data mining techniques. *Expert Syst. Appl.* 39, 10 (2012), 8707–8717. Navigate to

[38] Dusan Stevanovic, Natalija Vlajic, and Aijun An. 2013. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Appl. Soft Comput.* 13, 1 (2013), 698–708. Navigate to

[39] Antoine Vastel, Walter Rudametkin, Romain Rouvoy, and Xavier Blanc. 2020. FP-Crawlers: Studying the resilience of browser fingerprinting to block crawlers. In *Proceedings of the NDSS Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb'20)*. Navigate to

[40] Luis Von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. 2003. CAPTCHA: Using hard AI problems for security. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 294–311. Navigate to

[41] Gang Wang, Tristan Konolige, Christo Wilson, Xiao Wang, Haitao Zheng, and Ben Y. Zhao. 2013. You are how you click: Clickstream analysis for sybil detection. In *Proceedings of the USENIX Security Symposium*, Vol. 9. 1–008. Navigate to

[42] Ang Wei, Yuxuan Zhao, and Zhongmin Cai. 2019. A deep learning approach to web bot detection using mouse behavioral biometrics. In *Proceedings of the 14th Chinese Conference on Biometric Recognition (CCBR'19)*. 388–395. DOI: **https://doi.org/10.1007/978-3-030-31456-9_43 (https://doi.org/10.1007/978-3-030-31456-9_43)** Navigate to

[43] Mahdieh Zabihimayvan, Reza Sadeghi, H. Nathan Rude, and Derek Doran. 2017. A soft computing approach for benign and malicious web robot detection. *Expert Syst. Appl.* 87 (2017), 129--140. DOI: **https://doi.org/10.1016/j.eswa.2017.06.004 (https://doi.org/10.1016/j.eswa.2017.06.004)** Navigate to

Footnotes

[1] **https://www.seleniumhq.org/ (https://www.seleniumhq.org/)**.

[2] **https://developers.google.com/recaptcha/docs/display (https://developers.google.com/recaptcha/docs/display)**.

[3] **https://www.google.com/recaptcha (https://www.google.com/recaptcha)**.

[4] Please note that alternative approaches for the collection of mouse movements can be followed (such as the use of HTML div tags that utilise css hover selectors to request a new background image when visitors move their mouse over a box on the grid) if we want to avoid using JavaScript on the client side for the collection of mouse movements. This should not affect the detection flow of the framework.

[5] **https://scikit-learn.org/stable/ (https://scikit-learn.org/stable/)**.

[6] **https://keras.io/ (https://keras.io/)**.

[7] **https://www.wikipedia.org/ (https://www.wikipedia.org/)**.

[8] **https://keras.io/getting_started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development (https://keras.io/getting_started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development)**.

[9] **https://www.seleniumhq.org/ (https://www.seleniumhq.org/)**.

This work was supported by the FORESIGHT (H2020-833673), ECHO (H2020-830943), and IDEAL-CITIES (H2020-778229) projects, funded by the European Commission.

Authors' addresses: C. Iliou, Information Technologies Institute, CERTH, Thessaloniki, Greece, and BU-CERT, Bournemouth University, Bournemouth, United Kingdom; email: **iliouchristos@iti.gr (mailto:iliouchristos@iti.gr)**; T. Kostoulas, Department of Information & Communication Systems Engineering, University of the Aegean, Samos, Greece, and Department of Computing and Informatics, Bournemouth University, Bournemouth, United Kingdom; email: **theodoros.kostoulas@aegean.gr (mailto:theodoros.kostoulas@aegean.gr)**; T. Tsikrika, S. Vrochidis, I. Kompatsiaris, Information Technologies Institute, CERTH, Thessaloniki, Greece; emails: **theodora.tsikrika@iti.gr (mailto:theodora.tsikrika@iti.gr)**, **stefanos@iti.gr (mailto:stefanos@iti.gr)**, **ikom@iti.gr (mailto:ikom@iti.gr)**; V. Katos, BU-CERT, Bournemouth University, Bournemouth, United Kingdom; email: **vkatos@bournemouth.ac.uk (mailto:vkatos@bournemouth.ac.uk)**.

2576-5337/2021/06-ART24
DOI: **https://doi.org/10.1145/3447815 (https://doi.org/10.1145/3447815)**

Publication History: Received February 2020; revised December 2020; accepted January 2021