



**Bournemouth
University**

FACULTY OF SCIENCE & TECHNOLOGY

Dissertation submitted for the degree of

Doctor of Philosophy

February 2022

**Machine Learning Based Detection and Evasion
Techniques for Advanced Web Bots**

by

Christos Iliou

Copyright Statement

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright rests with its author and due acknowledgement must always be made of the use of any material contained in, or derived from, this thesis.

Abstract

Web bots are programs that can be used to browse the web and perform different types of automated actions, both benign and malicious. Such web bots vary in sophistication based on their purpose, ranging from simple automated scripts to advanced web bots that have a browser fingerprint and exhibit a humanlike behaviour. Advanced web bots are especially appealing to malicious web bot creators, due to their browserlike fingerprint and humanlike behaviour which reduce their detectability.

Several effective behaviour-based web bot detection techniques have been proposed in literature. However, the performance of these detection techniques when targeting malicious web bots that try to evade detection has not been examined in depth. Such evasive web bot behaviour is achieved by different techniques, including simple heuristics and statistical distributions, or more advanced machine learning based techniques. Motivated by the above, in this thesis we research novel web bot detection techniques and how effective these are against evasive web bots that try to evade detection using, among others, recent advances in machine learning.

To this end, we initially evaluate state-of-the-art web bot detection techniques against web bots of different sophistication levels and show that, while the existing approaches achieve very high performance in general, such approaches are not very effective when faced with only advanced web bots that try to remain undetected. Thus, we propose a novel web bot detection framework that can be used to detect effectively bots of varying levels of sophistication, including advanced web bots. This framework comprises and combines two detection modules: (i) a detection module that extracts several features from web logs and uses them as input to several well-known machine learning algorithms, and (ii) a detection module that uses mouse trajectories as input to Convolutional Neural Networks (CNNs).

Moreover, we examine the case where advanced web bots utilise themselves the recent advances in machine learning to evade detection. Specifically, we propose two novel evasive advanced web bot types: (i) the web bots that use Reinforcement Learning (RL) to update their browsing behaviour based on whether they have been detected or not, and (ii) the web bots that have in their possession several data from human behaviours

and use them as input to Generative Adversarial Networks (GANs) to generate images of humanlike mouse trajectories. We show that both approaches increase the evasiveness of the web bots by reducing the performance of the detection framework utilised in each case.

We conclude that malicious web bots can exhibit high sophistication levels and combine different techniques that increase their evasiveness. Even though web bot detection frameworks can combine different methods to effectively detect such bots, web bots can update their behaviours using, among other, recent advances in machine learning to increase their evasiveness. Thus, the detection techniques should be continuously updated to keep up with new techniques introduced by malicious web bots to evade detection.

Keywords: web bots, web bot detection, evasive web bots, advanced web bots, mouse movements, mouse biometrics, humanlike behaviour, machine learning, convolutional neural networks, generative adversarial networks, reinforcement learning

Dissertation Declaration

I agree that, should the University wish to retain it for reference purposes, a copy of my dissertation may be held by Bournemouth University normally for a period of 3 academic years. I understand that once the retention period has expired my dissertation will be destroyed.

Confidentiality

I confirm that this dissertation does not contain information of a commercial or confidential nature or include personal information other than that which would normally be in the public domain unless the relevant permissions have been obtained. In particular any information which identifies a particular individual's religious or political beliefs, information relating to their health, ethnicity, criminal history or sex life has been anonymised unless permission has been granted for its publication from the person to whom it relates.

Requests for Information

I agree that this dissertation may be made available as the result of a request for information under the Freedom of Information Act.

Signed: _____

Name: Christos Iliou

Date:

Programme: Doctor of Philosophy - Faculty of Science and Technology

Original Work Declaration

This dissertation and the project that it is based on are my own work, except where stated, in accordance with University regulations.

Signed: _____

Name: Christos Iliou

Date:

Acknowledgements

I would like to express the deepest appreciation to my professor, Vasilis Katos, for his crucial guidance and support throughout the completion of this thesis. Additionally, I would like to thank my supervisory team, Dr. Ioannis Kompatsiaris and professor Theodoros Kostoulas, for their support and guidance throughout this research, essential for the completion of this thesis. Furthermore, I would also like to thank Dr. Stefanos Vrochidis and Dr. Theodora Tsikrika, whose help was indispensable for the completion of my work. Last but not least, I would like to thank my family and my partner for their support, which was vital for me to stay focused on my work.

Contents

1	Introduction	1
1.1	Motivation	1
1.1.1	Web Bots	1
1.1.2	Web Bot Detection	2
1.1.3	Advanced Evasive Web Bots	3
1.1.4	The Interplay Between Web Bot Detection and Detection Evasion	4
1.2	Aims and Objectives	5
1.3	Research Methodology	6
1.4	Novel Aspects and Contribution	8
1.5	Outline of Thesis	11
2	Literature Review	14
2.1	Web Bot Landscape	14
2.1.1	Malicious Web Bots Functionality	16
2.1.2	Sophistication Levels	19
2.2	Web Bot Detection Techniques	22
2.2.1	Signature Based Detection	23
2.2.2	Behaviour Based Detection	26
2.3	Detection Evasion Techniques	31
2.3.1	Evasion of Signature Based Detection Techniques	32
2.3.2	Evasion of Behaviour Based Detection Techniques	34
2.4	Conclusions, Insights, and Limitations	38
3	Behaviour Based Web Bot Detection	41
3.1	Machine Learning Based Detection Using Web Logs	42
3.1.1	Web Bot Detection Framework	43
3.1.2	Evaluation	47
3.1.3	Results	53
3.1.4	Discussion	60

3.2	Machine Learning Based Detection by Combining Web Logs with Mouse	
	Behavioural Biometrics	62
3.2.1	Web Bot Detection Framework	62
3.2.2	Classification Methods for Web Bot Detection	69
3.2.3	Evaluation	70
3.2.4	Results	83
3.2.5	Discussion	90
4	Advanced Evasive Techniques for Web Bots	93
4.1	Evasive Web Bots Using Deep Reinforcement Learning	95
4.1.1	RL Main Concepts for the Web Bot Detection/Evasion Problem	96
4.1.2	Evasive Web Bots Using RL	97
4.1.3	Evaluation	104
4.1.4	Results	109
4.2	Evasive Web Bots Using Generative Adversarial Networks	113
4.2.1	Detection Framework	114
4.2.2	Evasive Web Bots Using GANs	116
4.2.3	Evaluation	117
4.2.4	Results	121
5	Discussion and Reflection	124
5.1	Discussion	124
5.2	Evaluation of Aims and Objectives	126
6	Conclusions and Future Work	129
	Appendix A Features Proposed in Literature for Web Logs	143
	Appendix B Methods and Performance of Literature	149
	Appendix C Features for Web Logs Used	156
	Appendix D Ethics	158

List of Figures

1.1	The web bot detection/evasion problem, where both the web bots and the detection framework continuously update their techniques	5
1.2	General research methodology and novel aspects, using red colouring and underlining to indicate our work	9
1.3	Organisation of this report	13
2.1	Web bot landscape	15
2.2	Classification of articles based on web bot sophistication and the detection method followed	27
2.3	Web bot detection evasion techniques	34
3.1	The web bot detection framework	44
3.2	Automatic annotation process	46
3.3	Cumulative variance of PCA's components for simple (D1) and advanced (D2) web bots	55
3.4	The absolute mean value of each feature contribution to PCA components for the simple (D1) and advanced (D2) web bots	56
3.5	ROC curve of the Voting classifier for the simple (D1) and advanced (D2) web bots	58
3.6	Comparison of the effectiveness of the classification algorithms for humans (class 0) and web bots (class 1) for the D1 (simple web bots) and D2 (advanced web bots) datasets in the working point with FPR=0.01	59
3.7	Web bot detection framework architecture	63
3.8	Web bot detection module that uses web logs	64
3.9	Web bot detection module that uses mouse movements	66
3.10	Mouse movement collection process	67
3.11	Distribution of total requests per session for the first phase of experiments	73
3.12	Distribution of the total requests per session for the second phase of experiments	75
3.13	Sequential Forward Floating Selection for D3 (left) and D4 (right)	77

3.14 Performance over requests for D3 (left) and D4 (right)	86
3.15 Performance over requests for D5 (left) and D6 (right)	89
4.1 RL concepts	97
4.2 Web bot detection framework	99
4.3 Performance of the web bot detection framework	109
4.4 Performance of the web bot detection framework	112
4.5 Web bot detection framework	115
4.6 Evasive web bots	116
B.1 Classification of articles based on web bot sophistication and the detection method followed	155

List of Tables

3.1	Unique agent names and IPs	49
3.2	Human and Web bot sessions	49
3.3	The parameters used on the classification algorithms.	54
3.4	Ranked features using χ^2 for simple (D1) and advanced (D2) web bots . . .	55
3.5	Ranked and selected (bold and in brackets) features using SFS for simple (D1) and advanced (D2) web bots for each classification algorithm	57
3.6	Confusion matrix for advanced web bots (FPR=0.4, t=0.17)	58
3.7	Confusion matrix for advanced web bots (FPR=0.01, t=0.31)	59
3.8	Architecture of the network for the detection of web bots from mouse movements	70
3.9	Human, moderate and advanced web bots sessions and total requests for all sessions of the first phase of experiments (sessions/requests).	74
3.10	Human, moderate and advanced web bots sessions and total requests for all sessions of the second phase of experiments.	76
3.11	The parameters of classification algorithms that use web logs.	78
3.12	Browsing behaviour of human, moderate and advanced web bots.	80
3.13	Evaluation of the web bot detection framework per session for D3 and D4 .	84
4.1	RL concepts and instantiation into the web bot detection/evasion problem .	98
4.2	Rewards	107
4.3	Parameters and configurations	108
4.4	Performance of evasive web bots using heuristics	110
4.5	Evasion percentage (%) of RL bots with different deep learning architectures	110
4.6	Performance of evasive web bots on the re-trained detection server using the N3 architecture	112
4.7	CNN architecture for web bot detection	115
4.8	Generator Architecture	118
4.9	Discriminator Architecture	119
4.10	Users, Sessions, and images for each dataset	120

4.11 Performance of the detection framework	121
4.12 Performance of evasive web bots	122
4.13 Selected images from humans and generated by GANs	123
5.1 Mapping of the objectives to publications and chapters of this thesis	128
B.1 Performance of behaviour-based web bot detection techniques that use web logs. Red color is used to highlight very low performances	149
B.2 Performance of behaviour-based web bot detection techniques that use mouse trajectories of visitors	154

Acronyms & Abbreviations

Term	Description
AI	Artificial Intelligence
ADTree	Alternating Decision Tree
API	Application Programming Interface
ART	Adaptive Resonance Theory
AUC	Area Under Curve
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CART	Classification And Regression Trees
CNN	Convolutional Neural Network
CSC	Card Security Code
CVC	Card Validation Code
CV2	Card Verification Value
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCGAN	Deep Convolutional Generative Adversarial Network
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
DQN	Deep Q-Network
DTMC	Discrete-time Markov chain
FPR	False Positive Rate
FRS	Fuzzy Rough Set
GAN	Generative Adversarial Network
GBDT	Gradient Boosting Decision Tree Algorithm
GPCM	Granded Possibilistic c Means
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol

IoC	Indicator of Compromise
kNN	k-Nearest Neighbours
LSTM	Long Short Term Memory
MCL	Markov Clustering
ML	Machine Learning
MLP	MultiLayer Perceptron
NN	Neural Network
OCR	Optical Character Recognition
OS	Operating System
PCA	Principal Component Analysis
PSO	Particle Swarm Optimization
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RF	Random Forest
RIPPER	Repeated Incremental Pruning to Produce Error Reduction
RL	Reinforcement learning
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROI	Return on Investment
SEO	Search Engine Optimisation
SFFS	Sequential Forward Floating Selection
SFS	Sequential Feature Selection
SOM	Self-Organizing Map
SVM	Support Vector Machine
TD	Temporal Difference
UI	User Interface
URI	Uniform Resource Identifier

1 Introduction

1.1 Motivation

1.1.1 Web Bots

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. In particular, they are responsible for numerous browsing automation processes, such as web indexing, website monitoring and testing (validation of hyperlinks, HTML code, and site functionality), 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 Imperva (2021), web bots accounted for 40.8% of the total traffic that they monitored.

Early versions of web bots were simple scripts. However, the need from web bots to perform more advanced automation tasks, including performing mouse movements and keystrokes, resulted in the creation of browsing automation software. This software can be used to perform actions similar to human visitors, such as mouse over actions, clicking on items, filling forms, etc. Two of the most well known browsing automation software are Selenium¹ and Puppeteer.² Both tools support the main functionalities of a browser, including mouse movements, clicks, and keystroke actions.

Web bots that are based on browsing automation software can be configured to have a fingerprint that is very close to a browser one (i.e., they cannot be distinguished from browsers using special scripts that collect hardware and software information about the device, which would normally reveal the browsing automation software). For example, the Puppeteer stealth plugin³ has been designed to make Puppeteer browsing automation software harder to detect. Additionally, web bots can be configured to exhibit a humanlike browsing behaviour. Thus, such web bots are very appealing to malicious authors that aim at creating web bots that evade detection by presenting themselves as humans.

¹<https://www.seleniumhq.org/>

²<https://pptr.dev/>

³<https://www.npmjs.com/package/puppeteer-extra-plugin-stealth>

Such evasive web bots can be and have been abused for malicious purposes, since they can avoid detection. Distil Networks (2018) has shown that 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. This makes them particularly dangerous since they present themselves as humans and perform several actions in a humanlike way, which severely hinders their detectability.

Allowing malicious web bots to visit a web server and perform uncontrolled actions can have devastating results. As discussed in different works in literature, such as Watson and Zaw (2018), Distil Networks (2019) and Azad et al. (2020), malicious web bots can perform a variety of malicious behaviours, including 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, generating accounts to spam messages or amplify propaganda, and more.

1.1.2 Web Bot Detection

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 to limit those malicious acts.

For many years, the most popular techniques for detecting web bots were based on CAPTCHAs (i.e., Completely Automated Public Turing test to tell Computers and Humans Apart) proposed by von Ahn et al. (2003). 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 extraction of letters from a distorted image or audio file (such as Google's reCAPTCHA version 1), and the click of a checkbox on a web page which, in some cases, also includes the selection of images which fulfil certain criteria (such as Google's reCAPTCHA version 2⁴).

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

⁴<https://developers.google.com/recaptcha/docs/display>

popular CAPTCHA challenges, ranging from simply using public online speech to text engines proposed by Bock et al. (2017), to a combination of several techniques, including image reverse search, tagging, recognition, and processing as shown by Sivakorn et al. (2016). 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). Additionally, the latest version of Google's CAPTCHA challenge⁵ (reCAPTCHA version 3), introduced in 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 in literature propose the use of (i) rule-based web bot detection based on browser fingerprinting techniques, and (ii) web bot detection based on the behaviour of the visitors (e.g., the spatial characteristics of the mouse movements, the browsing speed, etc.). For the latter, several measurable values (i.e., features) are extracted from the visitors' behaviour and are used as input to machine learning models to distinguish human behaviours from web bots' ones. In a similar manner, commercial solutions, such as Distil Networks (2019), combine the rule-based browser fingerprinting techniques with behaviour-based detection techniques to more effectively detect web bots.

After a visitor is identified as a bot by the aforementioned approaches, additional steps are taken, which can also be chosen by the administrators of the sites. Based on Distil Networks (2019), such actions include delivering different content to bots, or requesting from the visitor to prove that they are human by solving additional challenges (e.g., visual challenges).

1.1.3 Advanced Evasive Web Bots

Evasive web bots use different techniques to evade detection; such techniques can also be specific to the detection mechanism(s) that they face. Advanced web bots use the aforementioned challenge-solving capabilities (e.g., CAPTCHA-solving software) when

⁵<https://www.google.com/recaptcha>

necessary, but also increase their evasiveness by introducing additional evasion techniques based on the current state-of-the-art detection techniques that they may face: (i) the generation of a fingerprint similar to a browser one to evade detection based on their fingerprint, and (ii) the generation of a humanlike behaviour to evade detection based on their behaviour.

To achieve a fingerprint similar to a browser one, web bots can use specially configured browsing automation software that allows them to avoid detection techniques that examine their fingerprint, as mentioned in Laperdrix et al. (2020) and Jonker et al. (2019). Moreover, plugins that increase the evasiveness of such software against detection based on their fingerprint are available, such as the Puppeteer stealth plugin. Additionally, web bots can also use regular browsers (instead of using an automated browsing software), which make fingerprint based detection even harder, as shown by Akrouf et al. (2019). Thus, in this thesis we chose to focus on behaviour-based methods for the detection of advanced web bots.

Concerning the behaviour, web bots can use heuristics, statistical distributions, or more advanced machine learning based techniques to generate a humanlike behaviour. For example, Iliou et al. (2017) and Acien et al. (2020b) proposed the use of statistical distributions to simulate a humanlike browsing behaviour. Additionally, Akrouf et al. (2019) proposed the use of Reinforcement Learning (RL) and Acien et al. (2020b) proposed the use of Generative Adversarial Networks (GANs) to create evasive web bots.

Additionally, as discussed in Distil Networks (2019), there has been a rise in the sophistication of evasive web bots over the years, especially concerning their behaviour. Advanced web bots started exhibiting a sophisticated behaviour that is close to humans, making it difficult to be recognised as bots. Since such characteristic can be useful for malicious web bots, there is a need by web bot detection frameworks to identify such sophisticated bots. However, this is something that has not been examined in literature, where bots are not split based on their sophistication. Thus, even though web bot detection methods in literature achieve very high performance, it is not evident how effective they are against only advanced web bots, which can be particularly dangerous, as discussed before.

1.1.4 The Interplay Between Web Bot Detection and Detection Evasion

As shown in Figure 1.1, both parties (i.e., the evasive web bots, and the web bot detection frameworks) continuously update their techniques to achieve their goals. Web bots

continuously try to find ways to evade detection, and, at the same time, web bot detection frameworks continuously update their models and techniques to detect new evasive techniques introduced by web bots.

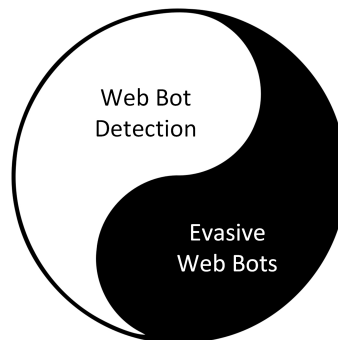


Figure 1.1: The web bot detection/evasion problem, where both the web bots and the detection framework continuously update their techniques

To this end, in this thesis the web bot detection problem is examined from both the defender's side (i.e., the web server that deploys web bot detection techniques), as well as the attacker's side (i.e., the advanced web bots that try to remain undetected). We argue that using this approach will allow us to gain a better understanding of the web bot detection problem, as well as the detection mechanisms that can be and are used.

1.2 Aims and Objectives

This research aims to answer the following question:

- **R:** How (malicious) advanced web bots can be detected based on their behaviour

This question comprises and can be split into three more targeted questions that cover the different aspects of the main research question:

- **R1:** What are the different types of web bots based on their sophistication and functionality
- **R2:** What techniques can be used to detect web bots based on their behaviour, focusing on detecting advanced web bots
- **R3:** How effective are behaviour-based web bot detection techniques against advanced web bots that use sophisticated techniques to evade detection

Thus, the main aims of this thesis and the respective objectives for each aim are:

- **A1:** To outline the web bots' landscape, considering the bots' functionality, purpose, and sophistication
 - **O1.1:** Review of academic publications as well as company reports regarding the web bots landscape
 - **O1.2:** Categorisation of web bots and definition of the functionality of the advanced web bots that this research will focus on
- **A2:** To evaluate current web bot detection techniques, and propose novel techniques for the detection of advanced web bots based on their behaviour
 - **O2.1:** Creation/Usage of datasets for evaluation
 - **O2.2:** Evaluation of state-of-the-art behaviour-based web bot detection techniques against different types of web bots based on their sophistication (e.g., simple vs advanced)
 - **O2.3:** Proposition of a novel web bot detection framework to detect advanced web bots
- **A3:** To evaluate how well the aforementioned novel web bot detection techniques perform against advanced web bots that use the recent advances in machine learning to evade detection based on their behaviour
 - **O3.1:** Identification of different machine learning based techniques that web bots can use to generate an evasive behaviour
 - **O3.2:** Evaluation of the latest and most effective web bot detection techniques against the advanced web bots that use the aforementioned machine learning techniques to avoid detection

The evaluation of how our objectives have been achieved (including the chapters and publications corresponding to each objective) is presented in Section 5.2.

1.3 Research Methodology

The purpose of this work is to research techniques that can be used to detect advanced web bots based on their behaviour. However, very few works exist that analyse the landscape of malicious web bots and the techniques that they use to evade detection based on their behaviour. Also, there are different techniques that sophisticated web bots can

use to generate evasive behaviours, including those that take advantage of recent advances in machine learning and deep learning, that have not yet been examined in literature.

Thus, in this work we approach the web bot detection problem from both the defender's side (i.e., the web server that deploys web bot detection techniques), as well as the attacker's side (i.e., the advanced web bots that try to remain undetected). We argue that both parties constantly update their techniques to achieve their goal; the defender examines additional characteristics of the visitors and uses novel techniques for the detection of the web bots, while the web bots try to generate fingerprints and exhibit behaviours that are close to human visitors, allowing them to remain undetected.

We initially outlined the web bot landscape, defining the different types of web bots based on their functionality, and sophistication. This landscape was periodically updated during the duration of this research and was used as the base for this research. The different sophistication levels and functionality of web bots presented in literature indicate that web bots can vary in characteristics and behaviour, and that advanced web bots can be very close to human visitors.

At the same time, we investigated how state-of-the-art web bot detection techniques (at that time) performed against web bots of different sophistication levels. The state-of-the-art behaviour-based web bot detection techniques at that time were using only web logs to identify whether a visitor is a bot or a human. Thus, in our experiments we used the web logs of a public web server that was visited regularly by humans as well as (malicious) web bots. During our experiments we showed that state-of-the-art web bot detection approaches performed poorly in detecting advanced web bots that try to evade detection.

This motivated us to perform further research on the advanced web bot detection problem and to identify alternative solutions, including using the mouse trajectories of visitors to detect web bots. At that time, different researchers who were also working on the same topic also started publishing new approaches that use mouse trajectories for the detection of web bots. Thus, we combined our current framework and techniques with the newly published works of that time and proposed a novel web bot detection framework that comprises and advances two detection modules, one that uses web logs and one that uses mouse trajectories. To evaluate our novel framework, we created a dataset with several human subjects. The evaluation of our framework has shown that this combination of techniques outperforms the previous works.

In the last part of our research, we focused on the techniques that web bots can use to generate a behaviour that allows them to evade their detection. We opted to do that, since this can give additional insights on the web bot detection problem and the robustness of the current detection techniques against “active” adversaries. Specifically, we focused on web bots that may use the recent advances in machine learning to evade detection. For that, two cases were considered, (i) the case of web bots that update their browsing behaviour based on whether they have been detected or not, and (ii) the case of web bots having in their possession several data from human behaviours and using them to generate (synthetic) humanlike data that can be used to generate a humanlike behaviour. Based on the evaluation of these approaches we concluded that the web bot detection problem is not a static problem and that both parties (the detection framework, and the web bots) can continuously update their techniques to achieve their goals.

1.4 Novel Aspects and Contribution

This thesis examines the web bot detection problem from both the defender’s side (i.e., the web server that deploys web bot detection techniques), as well as the attacker’s side (i.e., the advanced web bots that try to remain undetected). Thus, in this thesis (i) we propose novel methods for detecting web bots that have been designed to overcome limitations of the approaches proposed in literature, and (ii) we examine novel techniques that web bots can use in their attempt to evade detection. We argue that using this approach will allow us to gain a better understanding of the web bot detection problem, as well as the detection mechanisms that can be and are used.

In Figure 1.2 we include a diagram summarising the different approaches and techniques that (to the best of our knowledge) have been proposed in literature (including our published works, with red colouring, and underlined). Specifically, we approach the behaviour-based web bot detection/evasion problem using three methodologies based on the data that are utilised: the “Web logs” (blue), the “Mouse trajectories” (green), and the “Combination” (purple). For each of those methodologies, different techniques have been used in literature, such as heuristic approaches that may use statistics, and different machine learning (ML) and deep learning (DL) algorithms.

In the early days, literature in the web bot detection that examined visitors behaviour used web logs. Different supervised and unsupervised machine and deep learning based techniques for the detection of web bots have been proposed and improved over the

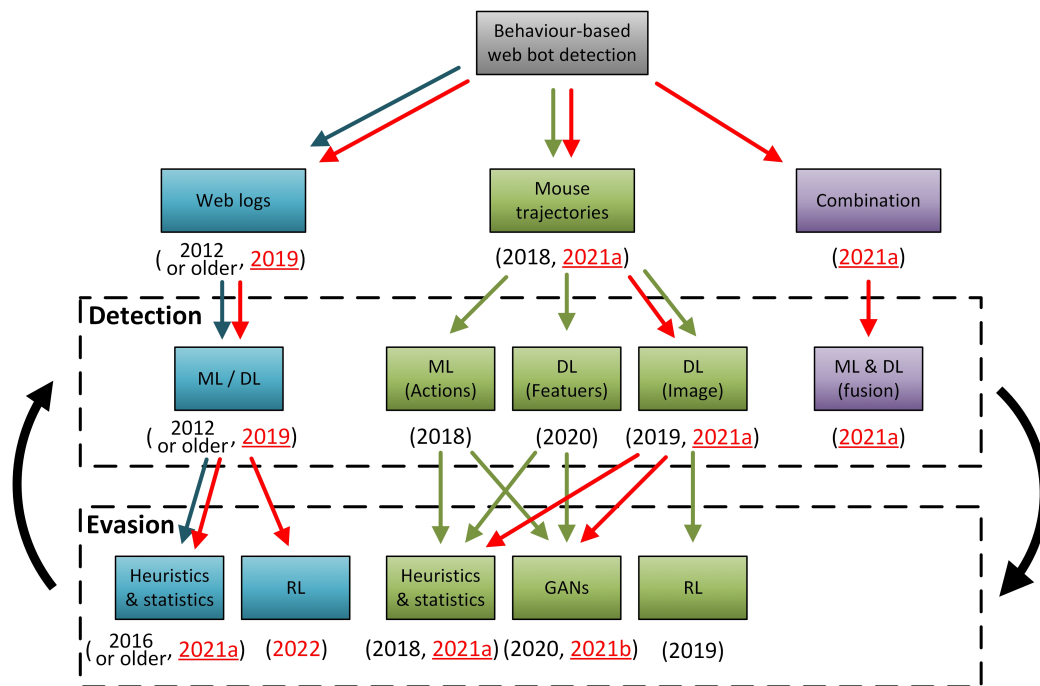


Figure 1.2: General research methodology and novel aspects, using red colouring and underlining to indicate our work

years. Additionally, techniques that are based on heuristics and statistical distributions were proposed to evade those detection techniques.

Later approaches proposed in literature used mouse trajectories for the detection of web bots. Similarly to web logs, different machine learning and deep learning algorithms were used. In this case, the approaches either extracted high level actions or features from the mouse trajectories of visitors, or processed the mouse trajectories as images. To evade those techniques, heuristics and statistical distributions were proposed (similarly to web logs), but also machine and deep learning techniques were used, including use of Reinforcement Learning (RL), and Generative Adversarial Networks (GANs).

This thesis has three main contributions. The first contribution of this research, published in Iliou et al. (2019), evaluates state-of-the-art (at that time) web bot detection techniques against web bots of different sophistication levels. Current approaches at that time focused on detecting web bots using web logs (see Figure 1.2, blue color). For that, in Iliou et al. (2019) we proposed a framework that uses state-of-the-art machine learning and deep learning techniques for the detection of web bots using web logs. This framework was evaluated on a real-world data collected from a public web server, containing both simple and advanced web bots.

Using this framework, we identified the unique challenges when state-of-the-art web

bot detection techniques are utilised for detecting advanced bots as opposed to simple bots. Specifically, the evaluation showed that very effective machine learning methods against simple web bots performed poorly when faced only with advanced web bots. Since advanced web bots require additional investment and effort on behalf of the malicious actors using them, their existence implies that higher impact attacks are performed with their use. Therefore, security solutions should give higher weight on detecting those advanced web bots.

The second contribution of this research published in Iliou et al. (2021a) addresses the aforementioned problem, by proposing a web bot detection framework that comprises two detection modules in a novel way: (i) a detection module that utilises web logs, and (ii) a detection module that leverages mouse movements. The evaluation of this framework showcases the effectiveness of this method in detecting advanced web bots. Additionally, in the same work a dataset was created that contained both the visitors web logs and mouse movements, and is available for sharing upon request.

Finally, the last contribution of this research focuses on the potential use of the recent advances in machine learning by web bots to evade detection. For that, we examined two novel cases on which machine learning techniques can be used for evasion: (i) advanced web bots updating their browsing behaviour based on whether they have been detected or not, and (ii) advanced web bots having in their possession several data from human behaviours and using them to generate (synthetic) humanlike data.

For the first case (i.e., bots using machine learning to update their browsing behaviour), we examined the case of advanced web bots using RL to evade detection by updating their browsing behaviour based on, among others, whether they have been detected or not. In Iliou et al. (2022) we show that, as opposed to approaches that use heuristics that usually get detected, RL can be used by web bots to learn browsing behaviours that can evade detection. To the best of our knowledge, this has not been examined in literature.

Concerning the case of web bots having been trained on human behaviours, in Iliou et al. (2021b) we evaluated the case where advanced web bots use GANs to generate images of trajectories similar to those of humans, which can then be used by bots to evade detection. We show that GANs are very effective in evading state-of-the-art detection techniques. At the time of publication, GANs had only been used by Acien et al. (2020b) in that area to generate synthetic swipe and accelerometer data, but not mouse trajectories.

The papers that present the outcomes of this research are listed below:

- **Iliou, C.**, Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, Y., 2019. Towards a framework for detecting advanced web bots. Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019, ACM, 18:1–18:10.
- **Iliou, C.**, Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2021. Detection of advanced web bots by combining web logs with mouse behavioural biometrics. *Digital Threats: Research and Practice*, 2 (3).
- **Iliou, C.**, Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2021. Web bot detection evasion using generative adversarial networks. IEEE International Conference on Cyber Security and Resilience, CSR 2021, Rhodes, Greece, July 26-28, 2021, IEEE, 115–120.
- **Iliou, C.**, Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2022. Web bot detection evasion using deep reinforcement learning. Proceedings of the 17th International Conference on Availability, Reliability and Security, ARES 2022, Vienna, Austria, August 23-26, 2022, ACM.

1.5 Outline of Thesis

In this section, the chapters of this thesis are outlined along with a brief overview of their contents.

In Chapter 2 we present the web bot landscape outlining the different types of web bots based on their functionality and sophistication (Section 2.1). Additionally, we include the literature review on the different signature-based and behaviour-based web bot detection techniques (Section 2.2). The behaviour-based detection techniques are split based on the two types of input sources considered in literature, i.e. the web logs, and the mouse movements. Moreover, we present the detection evasion methods that have been proposed in literature for the signature-based and behaviour-based web bot detection techniques (Section 2.3). Also, we conclude Chapter 2 with some insights on the current literature and limitations (Section 2.4)

In Chapter 3 we evaluate and show the limitations of common traditional web bot detection techniques against advanced web bots (Section 3.1). Additionally, we propose

a novel web bot detection framework that combines both web logs and mouse trajectories to overcome the limitations of the individual approaches when faced with advanced web bots (Section 3.2).

In Chapter 4 we evaluate the aforementioned web bot detection techniques against web bots that use recent advances in machine learning to evade detection based on their behaviour. Specifically, two approaches were examined; the web bots using RL to update their browsing behaviour based on whether they have been detected or not (Section 4.1), and the web bots using data of human mouse movements performed on web pages to learn and generate (synthetic) humanlike mouse trajectories using GANs (Section 4.2).

Finally, this thesis concludes with a discussion about the web bot detection and evasion problem and how our aims and objectives have been achieved (Chapter 5), followed by its conclusions and future work (Chapter 6).

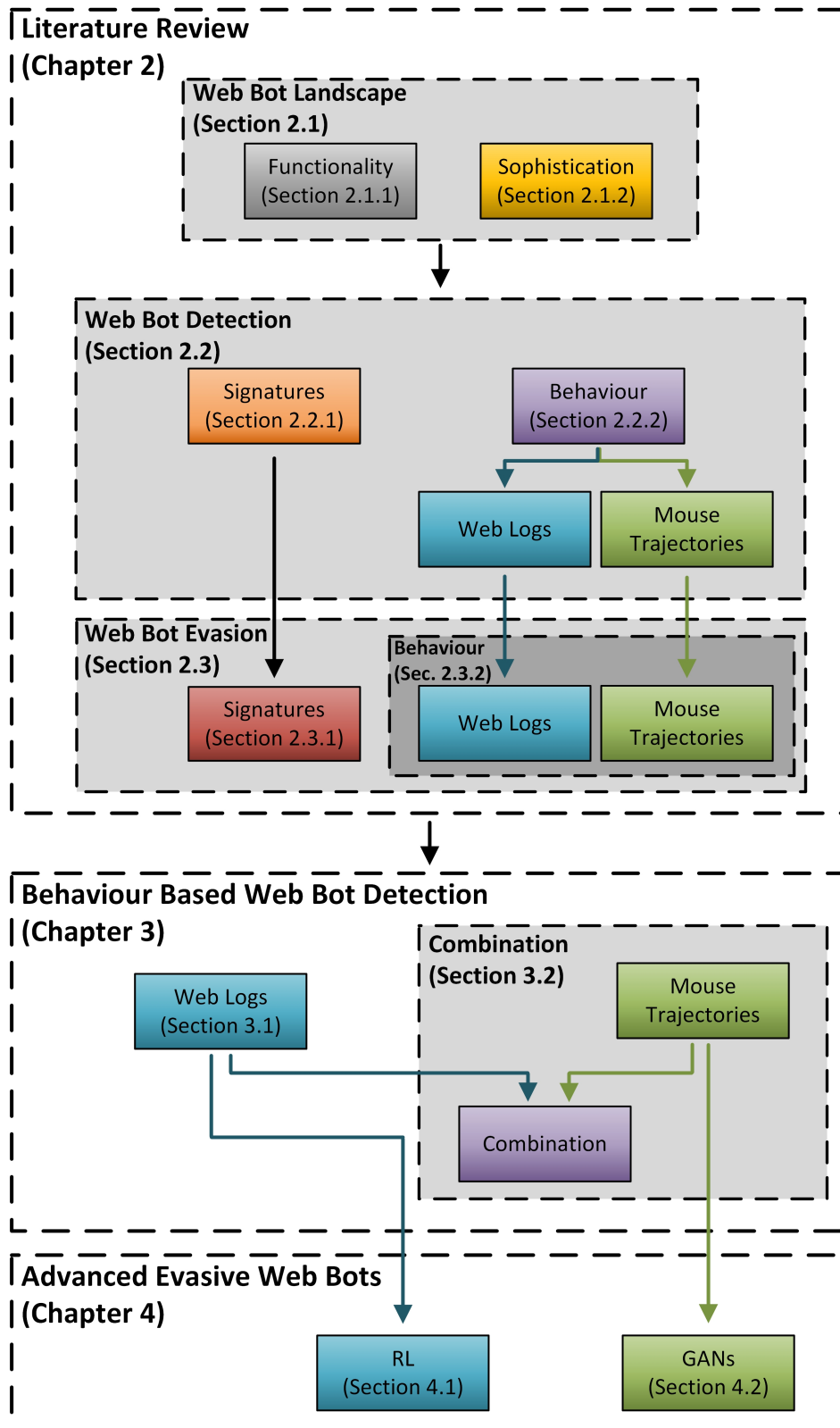


Figure 1.3: Organisation of this report

2 Literature Review

In this chapter we present the literature review on the web bot detection problem. Since there exist different types of web bots with different sophistication levels, we start by presenting the web bot landscape (Section 2.1). The landscape includes the different malicious actions that web bots can perform and showcases the importance of detecting such bots. Additionally, the landscape outlines the different sophistication levels that web bots need to have to perform the aforementioned actions, which shows the need of detection methods to consider those as well.

Then, we present the different web bot detection techniques that have been proposed in literature to detect different types of web bots based on their functionality and sophistication (Section 2.2). We show the drawbacks of signature-based detection techniques and the need for combining those with behaviour-based detection techniques, an approach that has been adapted by most well-known companies that offer web bot detection products and services.

Furthermore, since malicious web bots can use different techniques to evade detection, we also perform a literature review on those evasion techniques (Section 2.3). The latter can be used to better understand the current problems that new web bot detection approaches should focus on and address.

Finally, we conclude this chapter with some insights and limitations on the literature (Section 2.4). Also, we discuss the importance of the problem we aim to address in this thesis.

2.1 Web Bot Landscape

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. Based on their purpose, web bots vary in sophistication, as presented in Distil Networks (2019). 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 filling 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

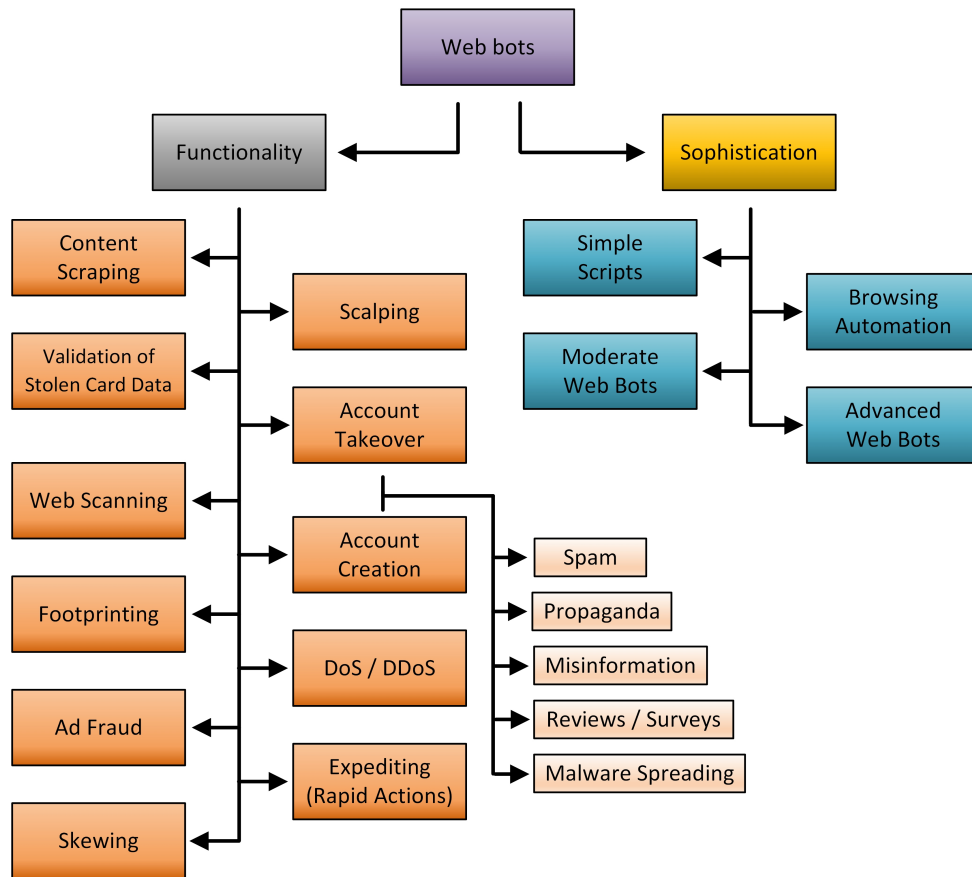


Figure 2.1: Web bot landscape

which supports all functionalities of a web browser should be used.

At the same time, as shown by the same report, Distil Networks (2019), such web bots may be abused for malicious purposes. These malicious web bots can perform highly-complex tasks and, at the same time, try to avoid detection by having a browser fingerprint and exhibiting a humanlike behaviour. This makes them particularly dangerous since they present themselves as humans and perform several actions in a humanlike way, which severely hinders their detectability. Distil Networks (2019) and Azad et al. (2020) have shown that malicious web bots are being used for several purposes such as price and content scraping, account takeover, credit card fraud, denial of service, and denial of inventory.

Next, we present a comprehensive landscape of the malicious web bots, including the different types of web bots based on their functionality (Section 2.1.1) and sophistication (Section 2.1.2) by reviewing different sources both from the academia as well as from companies that offer web bot detection products and extract analytics on their customer data. The main categories of web bots identified are presented in Figure 2.1.

2.1.1 Malicious Web Bots Functionality

Web bots allow the automation of different tasks, making them ideal for malicious actions that take place over the web. Below, we present a comprehensive list of the different types of malicious actions that have been presented in literature, to showcase the importance of detecting such bots.

Content Scraping

One of the most known usages of malicious web bots is content scraping. Distil Networks (2019) has shown that web bots can extract different types of information from a web page content depending on the sector that each web page belongs to (e.g., e-commerce, airlines, gambling, etc.). For example, web bots can gather information about the prices, the available inventory, information about the products and their availability (e.g., in the aviation domain, seat availability). Some of this information might require authentication, which is also achieved by web bots (as we will discuss in the following subsections). Additionally, Watson and Zaw (2018) show that scraper bots might try to access different paths and provide several parameter values for web pages and Application Programming Interfaces (APIs) to extract info and insights about the web pages structure.

Scalping

Another use of web bots, which has seen a rise in the past few years, is scalping; scalping bots that are used to obtain limited availability goods and services and usually resell them at higher prices. Such bots can monitor websites awaiting availability of goods and services and performs rapid actions depleting the available stock. These bots can use stolen credit cards to buy these products. This approach is called *cashing out* by Watson and Zaw (2018). Additionally, web bots might not complete the transaction but keep the products in their “inventory”, which may result in the denial of inventory as presented by Watson and Zaw (2018). As discussed in Distil Networks (2019), one of the targets of such bots are ticketing systems. Another well-known case is the GPU market, where, as also discussed in Molloy (2021), due to the limited stock of Nvidia’s new GPUs, scalper bots extinguished all the available stock to resell them at higher prices.

Credential Cracking / Account Takeover

Web bots have also been used for credential cracking purposes when web sites offer login functionalities. As shown by Watson and Zaw (2018), and Azad et al. (2020), common brute-force credential cracking techniques are used, such as password guessing, password spraying (i.e., trying a list of commonly used passwords against many different accounts), and credential stuffing (i.e., trying credentials obtained from breach dumps of unrelated accounts).

Account takeover is usually the first step of web bots to achieve a specific malicious purpose. Distil Networks (2019) and Imperva (2019) show that, after the account takeover, different actions can be performed. For example, in the aviation domain, web bots can use the accounts' loyalty program awards. Additionally, credit card and other account information may be accessible and retrieved by web bots, when they gain access to the account.

Account Creation

Automatic account creation can be used for different malicious purposes including content spam, amplifying propaganda (especially in social media), posting reviews, misinformation, and content at forums, filling surveys, and performing other similar actions depending on the site functionality. Also, hacked and malicious accounts can be used for spreading malware and skewing metrics calculated by the web sites. For the latter, sometimes there is no need to have/create an account (depending on the web site's functionality). Finally, having several accounts in their possession, attackers can obtain a consolidated overview of the application, and provide integrated reporting and analysis.

As also mentioned in Watson and Zaw (2018), web bots can use these fake accounts (as well as take advantage of web pages that do not require authentication) for Search Engine Optimisation (SEO), by making a website more visible to its target audience via boosting its ranking among the relevant results returned by mainstream search engines. That can be achieved by posting promotional content and referring links to websites on other websites. There are several tools called "black hat SEOs" that support bypassing security techniques commonly used by forums and blogs to deter automated spam, such as account registration, bot detection, many forms of CAPTCHAs, and e-mail activation before posting.

Validation of Stolen Card Data

Another common malicious action is when bots use payment pages of websites to validate in bulk stolen credit card data, gift cards, or codes such as coupons or vouchers. Such data can be obtained from illegal marketplaces from the dark web. Some applications allow brute-forcing missing payment information, such as expiration dates, Card Security Code (CSC), also referred to Card Validation Code (CVC), Card Verification Value (CV2), etc., when the other information has been stolen.

Web Scanning

Web bots can also be used for scanning web servers and applications for vulnerabilities. As discussed in Watson and Zaw (2018), web bots can identify paths, file names, parameters, perform fingerprinting actions to profile the application and identify versions of the software and frameworks used, etc.. This information can be used by attackers to identify weaknesses and potential vulnerable points to exploit. Vulnerable web servers can be used for different purposes including SEO, via, for example, defacing the websites as mentioned in Yang et al. (2021).

Footprinting

Footprinting techniques can be used to identify different properties of the application, such as composition, configurations, and security mechanisms to identify the underlying logic, structures, algorithms, functions, methods, etc. of application.

DoS / DDoS Attacks

Web bots have also been used for performing (Distributed) Denial of Service (DoS) attacks, such as HTTP-flooding attacks whose detection is also studied by Wang et al. (2015b). For that, web bots can perform specific functions that lead to the exhaustion of resources for the web pages, such as requesting several pages. As mentioned by Distil Networks (2019), this can be used by competitors to decrease the up-time and redundancy so that real customers cannot use the site, resulting in severe long-term effects on your online brand reputation and generate significant overage charges from hosting providers.

Ad Fraud

In a similar manner, web bots have been used for ad fraud. Such bots perform fake clicks and fraudulent display of web-placed advertisements. This falsification of the number of times an item such as an advert is clicked on, or is displayed can affect both the web page displaying this information as well as the customer wanting to advertise their products.

Expediting

Furthermore, there is a general category of bots called expediting bots that can be used for some of the aforementioned malicious purposes or for custom purposes (based on the web site) that perform rapid actions and violate explicit or implicit assumptions about the application's normal usage. As mentioned by Watson and Zaw (2018), this can give them unfair individual gain or cause loss to the other party. Similarly, actions by such bots can be performed "in the latest opportunity" on a website to achieve a specific objective and leaving insufficient time for another user to, for example, bid/offer (called *sniping*).

Skewing Decision-making Metrics

Finally, Distil Networks (2019) show that web bots can skew decision-making metrics by visiting links repeatedly, submitting specific forms, etc. This can affect specific application-based metrics such as visit counts, frequency of events, rate, etc. Some of that information can be visible to other users (e.g., visit counts) or affect the functionality (e.g., when dynamic pricing is provided). Additionally, this can affect investment decisions, as shown by Marenzi (2019), where it was estimated that 5% of all web traffic is attributable to investment-scraping bots.

2.1.2 Sophistication Levels

To achieve some of the aforementioned tasks, web bots should support several functionalities of a web browser, and, at the same time, exhibit sophisticated behaviours (i.e., perform tasks in a similar way as humans do). Next, we present the different sophistication levels of web bots showing that, based on the sophistication level, different detection techniques should be utilised.

Simple Scripts

These web bots usually use programming libraries that allow them to perform HTTP requests. They are usually simple scripts that do not exhibit any type of intelligence, perform multiple requests and follow a predictable behaviour. Such bots are used for very specific and simple tasks that do not require advanced functionality.

Since programming libraries are used, these bots have a fingerprint that is very different from a browser. Nikiforakis et al. (2013) have shown that the fingerprint of such bots can reveal their nature. Since they use such libraries, usually these bots have agent names that do not match any browser. Additionally, these libraries usually do not support main functionalities of the browser, such as parsing and running JavaScript, keeping cookies, performing any mouse movements, or pressing keystrokes.

Moreover, such web bots have behaviours that are very different from humans as shown by our work, Iliou et al. (2019). They follow consecutive hyperlinks on a web page with rapid speed. As a result, rapid browsing speed and browsing behaviour can indicate bot nature.

Web Bots Using Browsing Automation Software

As discussed before, simple scripts do not support functionalities of a browser, such as running JavaScript or keeping cookies. However, some (legitimate) purposes of web bots require those. Thus, the second category of web bots use browsing automation software that allows them to visit web pages and process them as a real browser.

Such bots use browsing software either in headless mode or by using the browsers' User Interface (UI) to visit the web pages. They do not have any intelligence and perform actions similar to simple bots. However, they can run JavaScript on web pages which may be useful when this is required to display the pages correctly.

As discussed in Azad et al. (2020), Laperdrix et al. (2020), and Schwarz et al. (2019), such software has a fingerprint that can be distinguished from that of common browsers. For example, as mentioned by Azad et al. (2020), browsing automation software, such as Selenium, can be detected by printing a stack trace and searching for the "selenium" keyword. Additionally, browsing automation software can be detected by examining specific JavaScript variables, such as a variable that starts with "cdc_" which is unique to Selenium. Similar variables can be found in other automation software.

Moderate Web Bots

Moderate web bots have a browser-like fingerprint and perform some additional actions such as mouse movements and keystrokes. Similarly to the previous category of web bots, moderate web bots use browsing automation software and exhibit a more advanced behaviour via keeping sessions, performing logins, adjusting their behaviour (e.g., adding random “sleep” time between successive requests).

Additionally, such web bots perform mouse movements and keystrokes, allowing them to perform more advanced tasks, such as testing the functionality of web applications similarly to humans. In some cases, they may try to mimic a humanlike behaviour using statistics.

Malicious moderate web bots can also use real browsers via malware installed into victims. Additionally, they may change IPs, use proxies, and combine different technologies to evade detection (depending on the web site).

Advanced Web Bots

We consider as advanced web bots the bots that have a browser-like fingerprint and exhibit a humanlike behaviour. These bots use specially configured browsing automation software or control directly the browsers allowing them to have fingerprints indistinguishable from real browsers. For that, anti-fingerprinting techniques can also be utilised, such as special applications that randomise properties of the browsers’ fingerprints, as shown by Schwarz et al. (2019). Such bots can also take advantage of plugins that increase the evasiveness of browsing automation software against detection based on fingerprint are available, such as the Puppeteer stealth plugin.

Additionally, such bots exhibit a humanlike behaviour in regards to how they navigate to web pages as well as the actions that they perform. They perform mouse movements, press keystrokes, fill in login forms, click on hyperlinks, and more.

Moreover, such bots can be equipped with CAPTCHA solving functionalities allowing them to bypass detection. As presented by Watson and Zaw (2018), web bots can utilise tools to perform Optical Character Recognition (OCR), or matching against a prepared database of pre-generated images, or using other machine reading, or human farms.

Also, as mentioned by Distil Networks (2019), advanced web bots can work as a globally distributed botnet. Such web bots use multiple IPs (which can be either region specific or global) and perform single-request attacks and user agent rotation, making

their detection even harder.

Finally, advanced web bots can take advantage of external services (using them as proxies) to increase their evasiveness by performing their requests through IPs of trusted organisations or IPs that have not shown any malicious activity before. For example, DataDome (2021b) shows the case where web bots used Facebook's link preview feature (and thus, the trusted/whitelisted Facebook infrastructure) to perform several requests and gather large amounts of data from the websites they were interested in.

2.2 Web Bot Detection Techniques

As presented above, it is of the utmost importance to detect web bots and apply the respective protection mechanisms. Web bot detection aims to accurately distinguish whether a web visitor is a bot or a human. This categorisation most commonly entails simply distinguishing web bots from human visitors, as it was done in Sisodia et al. (2015), Gabri et al. (2018), and Chu et al. (2018). Other works, such as Doran and Gokhale (2012), further categorise web bots based on their functionality, while Bai et al. (2014), Seyyar et al. (2017) and Zabihimayvan et al. (2017) categorise web bots based on their purpose.

Current web bot detection approaches used by commercial solutions, including the ones of the most well-known companies in this domain such as Distil Networks (2019), Akamai (2021), Cloudflare (2021), DataDome (2021a), and PerimeterX (2021), advertise that they combine (i) signature-based web bot detection techniques with (ii) web bot detection based on the behaviour of the visitors (e.g., the spatial characteristics of the mouse movements, the browsing speed, etc.). Similarly, in academia there are works that propose (i) signature-based techniques to identify web bots based on their fingerprint, such as Azad et al. (2020), Laperdrix et al. (2020), and Schwarz et al. (2019), and (ii) machine learning based techniques that examine the visitors' behaviour, such as Rovetta et al. (2020), Lagopoulos and Tsoumakas (2020), and Acien et al. (2020b 2021).

Finally, as mentioned in Distil Networks (2019), after a visitor is identified as a bot by Imperva's (former Distil) Advanced Bot Protection system, 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.

Next, we present in detail the two most prevalent methods for detecting web bots: (i) signature-based techniques that rely either on specific challenges or on examining different characteristics of the visitors to identify whether they are bots or not, and (ii) behavioural based techniques that examine the browsing behaviour of the visitor and usually train machine learning models to distinguish web bots from human visitors.

2.2.1 Signature Based Detection

As discussed in Azad et al. (2020), the most common signature based detection techniques can be split into three main categories: (i) visual challenges (e.g., CAPTCHA challenges), (ii) techniques that examine whether the visitor accesses specific resources that human visitors should not have accessed (e.g., because they are not visible when rendering the web page in a browser), and (iii) browser fingerprinting techniques.

Visual Challenges

For many years, the most popular techniques for detecting web bots were based on the work of von Ahn et al. (2003) on CAPTCHAs. 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, with most of them belonging to the general categories of text-based CAPTCHAs, image-based CAPTCHAs, and sound-based CAPTCHAs. Text-based CAPTCHA has been one of the most widely used types of CAPTCHA, as mentioned by Xu et al. (2020). In this type of CAPTCHA, users were required to recognise and extract letters and numbers from (distorted) images. Google's first version of CAPTCHA (reCAPTCHA version 1) also used this approach.

Image-based CAPTCHAs provide challenges that require the visitor to understand the image content (e.g., object identification, scene understanding, etc.) and select images from a grid of images which fulfil certain criteria. Google's second version of reCAPTCHA also uses this approach after the user behaviour analytics indicate bot behaviour. As mentioned in Xu et al. (2020), such approaches are also "mobile-friendly", since they simply require users to select images as opposed to writing text.

Audio-based CAPTCHAs are developed primarily for the visually impaired, acting as a replacement to text and image based CAPTCHAs. Visitors are requested to enter the words they hear spoken over an audio clip.

Finally, there have been some other not so well-known CAPTCHA-like challenges proposed in literature, such as “swipe” based challenges proposed by Jiang and Dogan (2018) for touch-enabled smart devices that ask users to move objects from the left side of the canvas to touch another specific object on the right side of the canvas.

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. These attacks led CAPTCHA challenges to increase in difficulty. Additionally, 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 behaviour-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). For example, the latest version of Google’s CAPTCHA (reCAPTCHA version 3), introduced in 2018, also runs in the background and generates a score that is based on interactions of the visitors. Such methods usually require as input both legitimate and abusive behaviours so as to be adjusted to each specific web site and work better.

Content Traps

One general and relatively old category of rule-based web bot detection techniques is the use of “crawling traps”, which in recent works such as Chen et al. (2020) is presented as honeypots. These techniques aim to distinguish humans from web bot visitors via creating special links that only web bots could follow. One reason for that is that humans see the web page as rendered by the browser, while web bots access the source code. Thus, adding special parts of the source code not visible to humans can lead web bots to access them and get detected.

There are several technical approaches to achieve that, such as including hyperlinks with the same colour as the background of the image, using CSS functionality to hide part of the web page, and more. For example, in Vastel et al. (2020) they detected crawler traps that used invisible links with the “nofollow” property that also appended a unique random identifier to the URL pointed to by the link.

Additionally, crawler traps can be created via the automatic generation of several web pages from a general link but with different parameters, as presented by David et al.

(2021). The latter can also lead to an infinite loop of links that the bot will follow. However, as shown by David et al. (2021) advanced techniques that utilise machine learning can be used to identify such traps.

Browser Fingerprinting

Finally, one of the more recent and highly effective signature-based web bot detection techniques is browser fingerprinting. Browser fingerprinting is a category of device fingerprinting or machine fingerprinting techniques, where the server collects information about the software and hardware of a remote computing device from the browser of the device.

Such techniques can very easily detect simple web bots and more sophisticated web bots that do not use advanced techniques to evade detection, and this is why they have been adopted by security companies as shown by Azad et al. (2020). There are several fingerprinting techniques that distinguish browsing automation software from real browsers. A comprehensive library that includes a considerable amount of those techniques is the FingerprintJS.¹ This library computes a hashed visitor identifier from them, which can be used to identify the same visitors even if the visitors use incognito/private mode and even when browser data is purged.

Techniques included in this library as well as additional techniques have been proposed in literature. As discussed in Azad et al. (2020), Laperdrix et al. (2020), and Schwarz et al. (2019), examples of such techniques include font detection, plugin enumeration, WebGL fingerprinting, examination of unique to browser automation software strings in JavaScript variables, and more. Furthermore, Schwarz et al. (2019) proposed more advanced fingerprinting techniques that can extract low level properties, such as the instruction-set architecture, and the memory allocator used by the browser.

As discussed in Section 2.1.2, web bots may use browsing automation software, with a fingerprint that is close to a browser one. Using the aforementioned advanced browser fingerprinting techniques, browsing automation software can be identified based on the fingerprint that it generates, even though it can be very close to a browser one.

However, the advances in the technologies used by web bots allow the creation of browsing automation software with a fingerprint that is very close to a browser one. As shown by Pastrana et al. (2018), Campobasso et al. (2019), and Sivakorn et al. (2016), a fingerprint that is close to one of a browser can be achieved with the use and configuration

¹<https://github.com/fingerprintjs/fingerprintjs>

of specific browsing automation software. Additionally, Akrouf et al. (2019) showed that web bots can also use regular browsers, which makes their fingerprint indistinguishable from common browsers. Thus, even though browser fingerprinting techniques are very effective in most cases, they can still be bypassed.

2.2.2 Behaviour Based Detection

Behaviour-based web bot detection techniques collect data generated from visitors while they are browsing the web server, and find unique characteristics from those data that can be used to distinguish bots from real visitors.

As opposed to signature-based detection techniques, detecting web bots based on their behaviour makes the evasion harder, since more effort is required to change the general behaviour of a tool as opposed to changing specific Indicators of Compromise (IoC) that it generates. Thus, as discussed in Bianco (2013), signature-based detection techniques that utilise different IoC have a low pain threshold (i.e., they require low effort to be changed).

To detect web bots based on their behaviour, methods proposed in literature typically examine: (i) the web logs that visitors generate, and (ii) the visitor's mouse movements. Figure 2.2 presents a classification of the articles based on the sophistication of the web bots in question (i.e., simple, moderate, or advanced) and the detection methods used (i.e., detection using web logs or mouse movements, and classification or clustering). Concerning the sophistication of web bots, when not mentioned explicitly by the authors of each work, we categorised the articles based on the labeling process performed. For example, when authors label a session based on the visitor's fingerprint (such as the agent name) or whether the visitor has accessed *robots.txt*, a file that is only meant to be accessed by web bots (and primarily good behaving web crawlers), we can assume that these are simple web bots.

As we can see, most works in literature focus on detecting simple to moderate web bots using web logs with methods that use classification algorithms. They usually address this problem as a binary classification problem, where the algorithm has to decide whether a visitor is a bot or not.

Additionally, we see that as web bots' sophistication increases, mouse trajectories are preferred, most probably because it is far more difficult for malicious users to create humanlike mouse movements as opposed to web logs. Also, simple web bots do not generate any mouse trajectories at all since, as discussed in Section 2.1.2, they usually

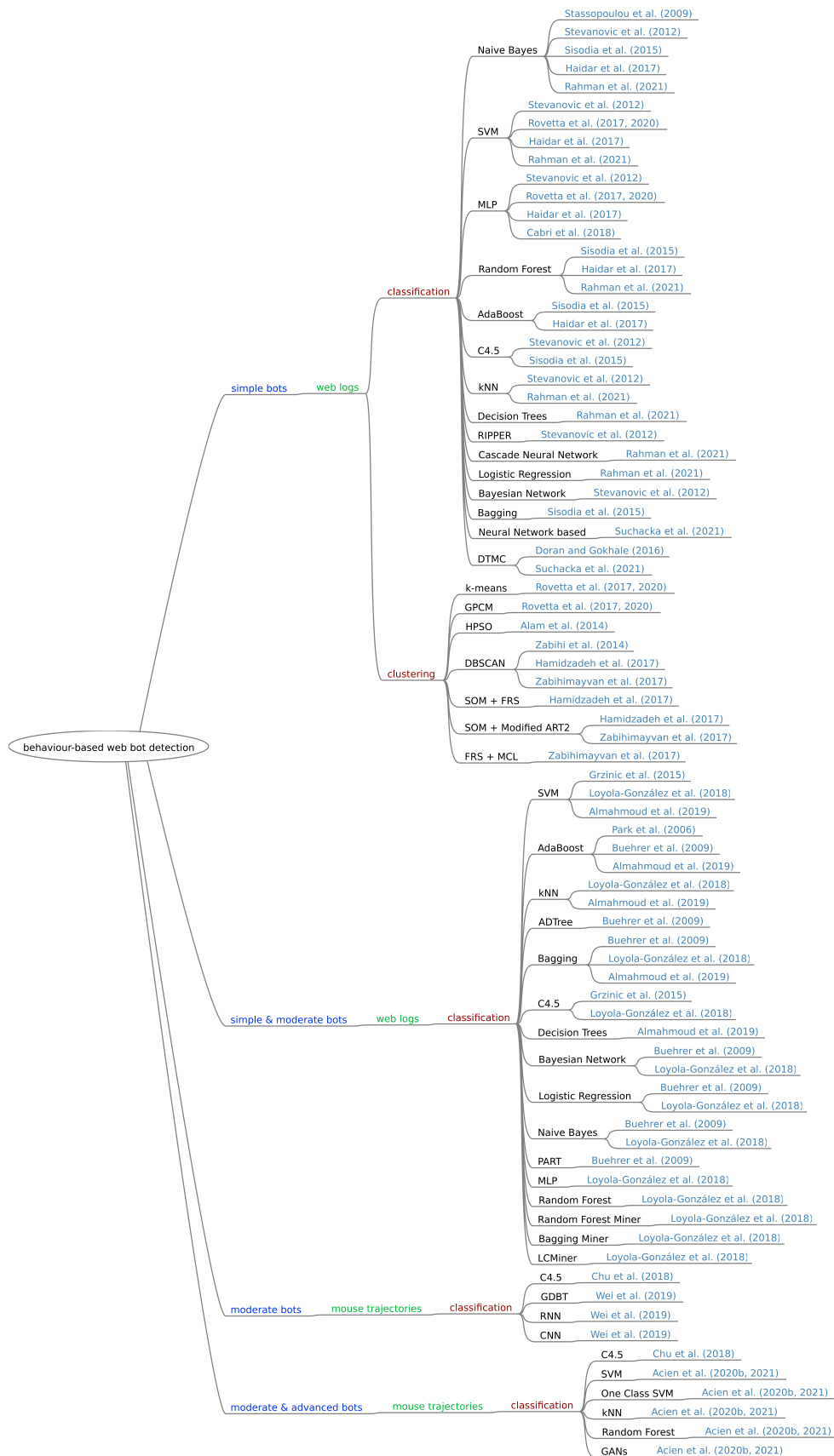


Figure 2.2: Classification of articles based on web bot sophistication and the detection method followed

use programming libraries that allow them to perform HTTP requests.

Next, we present in detail the literature on the behaviour-based web bot detection when web logs, and mouse movements are used.

Web Bot Detection Using Web Logs

The earliest approaches in behaviour-based web bot detection examine the web logs that the visitors make. Web logs are grouped into sessions and, based on these sessions, some measurable values (i.e., features) are extracted. These features are used to train machine learning models. The trained models are used to classify new visitors as bots or humans based on the web logs that they generated.

Since web logs of servers do not store information that can be used to uniquely identify different visitors by default (e.g., by the use of a PHP session ID), the majority of works in literature consider a combination of the IP with the browser agent name for the creation of a unique identifier per visitor. Examples of works that follow this approach are Stevanovic et al. (2012 2013), Bhargav and Bhargav (2014), Zabihi et al. (2014), Grzinic et al. (2015), Rude and Doran (2015), Sisodia et al. (2015), Doran and Gokhale (2016), Hamidzadeh et al. (2018), Rovetta et al. (2017 2020), and Suchacka et al. (2021). Only very few works, such as Haidar and Elbassuoni (2017) and Almahmoud et al. (2019), use a session ID value. However, as discussed in Section 2.1.2, in both cases (e.g., using the combination of IP with the agent name, and using a session ID) malicious users can manipulate their fingerprint by changing their agent name, IP, or removing the session cookies so as to be considered as a different user when they want to. There are more advanced techniques that can make this manipulation harder, such as advanced browser fingerprinting, discussed in Azad et al. (2020), Laperdrix et al. (2020), and Schwarz et al. (2019), but they are not widespread in literature yet, most probably because they require additional configurations of the web servers.

Then, to split the visitors' data into sessions, most works in literature propose the end of the session when more than 30 minutes have passed and no new requests with the same session identifier have been performed. Even though this is an arbitrary number, the proposed approaches that follow this approach achieved very high performance. Examples of works that follow this approach are Stevanovic et al. (2012 2013), Sisodia et al. (2015), Bhargav and Bhargav (2014), Zabihi et al. (2014), Rude and Doran (2015), Doran and Gokhale (2016), Hamidzadeh et al. (2018), Rovetta et al. (2017 2020) and Suchacka et al. (2021). Finally, Rude and Doran (2015) consider only sessions that

have a number of requests greater than a threshold, to make sure that there are enough data to distinguish web bots from humans.

After splitting the logs into sessions, several measurable values (i.e., features) are extracted. These features aim to describe the behaviour of the visitors based on the web logs that they generate and they have to do with the total requests that they make, the file types accessed, the values of the request headers, the URLs accessed, the time between successive requests, the semantics of the web pages accessed, and more. Also, these can be domain-specific (e.g., they are adapted to a specific domain such as the e-commerce domain), or generic ones that can be applied to any domain. A full list of features is presented in Appendix A. As we can see, these features try to represent the general behaviour of web bots and humans and, eventually, to detect web bots by distinguishing them. Also, as shown by Zabihimayvan and Doran (2018), some feature distributions of data collected from different web servers with different content and of different geographical regions share the same characteristics. However, this does not apply to all features. Thus, since most of these features alone can be simulated by evasive web bots when bots know the general behaviour of humans and the structure of the web servers of interest, the extracted features aim at covering different aspects of the visitors to make this harder.

Features extracted from web logs are used to train machine learning models to classify the new visitors as bots or humans. As shown in Figure 2.2 and Figure B.1 from Appendix, web bot detection approaches proposed in literature that examine web logs mainly use either classification or clustering algorithms for detection, with the former being considerably more popular. Also, we see that it is common for different works to use the same classification algorithms as opposed to clustering algorithms, where most of them are used by one or two works. For example, classification algorithms such as Naive Bayes, Support Vector Machines (SVM), MultiLayer Perceptron (MLP), Random Forest, and AdaBoost classifiers are very commonly used in the web bot detection using web logs. Additionally, as we see in Table B.1 from Appendix, there is no clear performance benefit of one algorithm against the others. We see that in some cases some algorithms work better while in others they do not. This, of course, depends on other aspects such as the dataset, extracted features, and experimental setup used in each work. The general performance of all algorithms is relatively high, except the case of Naive Bayes which has shown very low performance in Stevanovic et al. (2012) and Sisodia et al. (2015), which indicates that more focus should be given on the other aspects when detecting web bots

(such as the the general method and the feature engineering) rather than the algorithms themselves.

Furthermore, almost all works perform the detection process offline (i.e., after the end of the session), while only Rovetta et al. (2017) and Chu et al. (2018) performed it on-line by performing an estimation during the session outlining the importance of detecting web bots as soon as possible. However, this adds an overhead since several estimations should be made during a session of visitors. Additionally, less popular approaches extract patterns from the web requests and use these to detect web bots. Specifically, Kwon et al. (2012) extracted HTTP request patterns which considered whether the visitor accessed the “main” web page, other web pages, or files of a specific type (such as images, compressed files, binary executable files, etc.). Based on those, they created a table that shows which patterns appeared more often in human sessions and which in bot sessions. A new session is classified as bot or human in an online manner (i.e., as requests arrive) when a specific pattern is identified and based on how many such patterns were found in the training set for the human and bot visitors. In a similar manner, Doran and Gokhale (2016) extracted patterns and used them as input to two first-order discrete time Markov chain (DTMC) models, one for the humans and one for the bots. Again, based on which probability of those DTMC is higher for a specific pattern of a new session, the session is classified as human or bot. Even though these approaches work well, they aim at specific types of web bots (such as web crawlers, or image crawlers) that are primarily simple and do not download files like a browser does while visiting web pages (which is something commonly done by advanced web bots).

Web Bot Detection Using Mouse Trajectories

More recent research has proposed detection methods that use mouse movements. However, such approaches are relatively recent and there are only a few works that utilise mouse trajectories for the detection of web bots.

Chu et al. (2018) proposed the extraction of several high level actions from them (e.g., click, point-and-click, and drag-and-drop) and then extracted additional features from those actions, such as duration, distance, displacement, etc. These features are then used as input to machine learning algorithms in a similar manner as the features extracted from web logs. As discussed before, simulating humanlike mouse movements is far more difficult than web logs, thus features that have to do with mouse movements can be more effective in detecting web bots, especially those that try to evade detection.

Furthermore, Wei et al. (2019) utilised the very promising performance of Convolutional Neural Networks (CNNs) for identifying patterns (in this case humanlike and bot mouse movement patterns) from images, and proposed to transform the mouse trajectories into images and use them as input to CNNs.

Finally, Acien et al. (2021) focused on mobile web bots that try to generate touchscreen trajectories with similar features as the ones extracted from humans (such as duration of mouse movements, velocity, etc.) using statistics or GANs. By extracting features that have to do with the swipe duration, distance, displacement, angle, velocity, and efficiency and using them as input to traditional machine learning algorithms (i.e., SVM, kNN, and Random Forest), they were able to detect both types of web bots. However, the specific task (swipe action) is very limited, thus human and bot actions can be easily modelled.

As shown in Table B.2 from Appendix, web bot detection techniques that are based on mouse and touchscreen trajectories have shown very promising results when faced with web bots that try to present a humanlike behaviour through heuristic approaches. However, the difficulty in generating evasive web bots that have humanlike mouse movements outlines the limitations of the evaluation of the aforementioned methods. For example, Chu et al. (2018) have shown the effectiveness of such approaches against web bots that perform mouse movements following a specific type of lines heuristically generated. Additionally, Wei et al. (2019) tested these techniques against web bots that use similar types of lines generated heuristically, with parametric curves, or using statistical distributions. Finally, Acien et al. (2021) showed the effectiveness of those techniques against web bots that try to generate touchscreen swipe trajectories, and not generic touchscreen events while browsing. Thus, even though the aforementioned approaches have achieved very promising results, these techniques have not been tested with “more advanced” web bots.

2.3 Detection Evasion Techniques

As detection techniques advance, web bots update their methods to remain undetected. Web bots use and combine different techniques to evade signature-based detection as well as detection based on their behaviour. Next, we present the techniques that web bots can use to evade detection.

2.3.1 Evasion of Signature Based Detection Techniques

Visual Challenges

As discussed above, one of the earlier web bot detection techniques is the use of challenges to distinguish web bots from humans, with CAPTCHAs being the most common ones as mentioned by Na et al. (2020). Evasive web bots can be equipped with functionality that allows them to solve those challenges when requested.

For example, as shown by Chen et al. (2017), text-based CAPTCHAs (i.e., where an image is presented to visitors and they are requested to type the characters included into a box) can be solved using methods such as OCR techniques or CNNs. Even though some additional strengthening techniques have been proposed (e.g., addition of noise, image distortion, etc.), such techniques have also been bypassed by using OCR techniques or CNNs as shown by Chen et al. (2017). Additionally, Recurrent Neural Networks (RNNs) have also been used to solve such CAPTCHAs as shown by Rui et al. (2013).

The poor performance of text-based CAPTCHAs against the aforementioned attacks led to the use of different, image-based CAPTCHAs. These CAPTCHAs are challenges that request from users to select one or more images with specific semantic meanings from several candidate images. Thus, recognising the semantic of images is more difficult compared to character recognition, thus being more resilient to automated attacks. However, the advancements in machine learning and deep learning enabled the creation of techniques to bypass image-based CAPTCHAs. For example, Alqahtani and Alsulaiman (2020) proposed a method that employs deep learning technologies, and machine learning algorithms, including Random Forests, classification and regression trees (CART), bagging with CART, and Naïve Bayes, to automatically answer image-based CAPTCHA challenges. Similar techniques can be used by web bots to bypass those challenges.

Moreover, Sivakorn et al. (2016) proposed a combination of machine learning based techniques along with Google's reverse image search² and image tagging to bypass Google image-based reCAPTCHA. Specifically, to break Google's reCAPTCHA, Sivakorn et al. (2016) initially used Google's reverse image search to get more images and with higher resolution. Then, they used services such as the Clarifai³ using techniques from Zeiler et al. (2011) to return tags (along with confidence score) that describe the input images. Finally, since tags do not always match the description of reCAPTCHA, they used word

²<https://support.google.com/websearch/answer/1325808>

³<https://www.clarifai.com/>

embeddings using the Word2Vec algorithm, proposed in Mikolov et al. (2013), to calculate the similarity between the generated tags and the text of the reCAPTCHA.

Additionally, Bock et al. (2017) showed that Google's reCAPTCHA can also be bypassed easily by solving the audio challenge accompanying the CAPTCHA instead of the visual one. For that, they segmented the reCAPTCHA audio, and used it as input to several speech recognition tools (including Google Speech API, Bing Speech Recognition, and other similar services). Additionally, to address the issue of such services supporting general speech recognition (and not only digits), they also used two layers of phonetic mapping (i.e., exact-homophone, and near-homophone). Finally, they performed a voting over the output of all the speech recognition services.

Finally, web bots can be human assisted when solving those challenges. As mentioned in Datadome (2020), CAPTCHA farms are services that bot developers can use to use humans for solving such challenges. Thus, instead of using AI-based techniques, web bots distribute CAPTCHA challenges to a pool of human workers, usually in developing countries. When the challenges are solved, then web bots can continue to visit the web server undisturbed. Additionally, human farms can be used to generate a huge amount of such solved challenges to be used by AI for training.

Content Traps

Content traps can be evaded by programming web bots to examine some additional characteristics before visiting a hyperlink, such as the color or location of the hyperlink. As discussed above, crawler traps can use invisible links with the "nofollow" property and appends a unique random identifier to the URL pointed to by the link. These techniques are very effective but can be easily bypassed if the web bot authors know about them (or can identify them by examining the web pages of interest). Web authors can add specific rules that ensure that web bots do not follow such hyperlinks.

Browser Fingerprinting

Finally, as mentioned by Azad et al. (2020), web bots focus on generating a browser-like fingerprint to evade fingerprint based detection techniques, which are very common nowadays. For that evasive web bots use software that allows them to have fingerprint similar to browsers and support the majority of common browsers functionalities. To achieve that, Laperdrix et al. (2020), Bock et al. (2017), and Jonker et al. (2019) mention that web bots can use specially configured browsing automation software (since such

software in their vanilla configurations can be detected). Moreover, Akrouit et al. (2019) have shown that 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.

2.3.2 Evasion of Behaviour Based Detection Techniques

Web bots can use different techniques to exhibit a behaviour that can evade detection. Since behaviour-based bot detection methods examine visitors' web logs and mouse trajectories to decide whether they are bots or not, evasion methods proposed in literature focus on creating bots that exhibit a humanlike browsing behaviour regarding the web pages they visit, and the mouse movements that they generate. These approaches are summarised in Figure 2.3 and discussed below.



Figure 2.3: Web bot detection evasion techniques

Evasion of Behaviour Based Detection Techniques that Use Web Logs

Concerning the browsing behaviour of web bots (i.e., the web logs they generate), Yu et al. (2015) argues that there are three main aspects that have to be decided to model a humanlike behaviour: (a) the total number of web pages to go through, (b) the time to spend on each web page, and (c) the specific pages to visit. To model those, literature uses (i) approaches based on heuristics, (ii) statistical distributions, (iii) statistical models,

and (iv) machine learning. It is also common to face each aspect of the humanlike behaviour differently and to use different or combinations of the aforementioned techniques to simulate each aspect.

Approaches based on heuristics usually depend on several pre-defined parameters that are selected by the authors to generate a humanlike behaviour. For example, Pastrana et al. (2018) and Campobasso et al. (2019) proposed the addition of time sleeps on each web page simulating a “reading” functionality that depends on the length of text in each web page, and skipping or spending a few seconds (a value between two pre-defined integers indicating the seconds to “sleep”) on content that they have already visited. Additionally, Iliou et al. (2017) proposed the use of random sleeps (i.e., bots stop making requests for a specific period of time) that is also calculated based on some pre-defined values. Moreover, Pastrana et al. (2018) proposed the use of several pre-configured navigational patterns based on the types of web pages that they visit. These patterns can be used by web bots to present a humanlike browsing behaviour. However, as it is evident, deciding these navigational patterns requires expertise as well as knowledge of the target websites’ visitors behaviour and characteristics. Additionally, since visitors usually have different behaviours (even on the same web pages), it is not always feasible to model those patterns and it requires a lot of expertise. Campobasso et al. (2019) avoided this issue by enhancing the proposed web bots functionality by introducing a training period, where the bots “watch” a real human navigating the web pages and learn to repeat those actions. However, this also requires humans to navigate the web servers (at least during the bot training period).

Statistical distributions have also been used to model the different aspects of the humanlike behaviour. For that, researchers analyse the behaviour of several humans accessing web servers and see what behaviours match which statistical distributions. To model the total number of requests, Oikonomou and Mirkovic (2009) proposed the use of a uniform distribution, Yu et al. (2011) proposed the use of inverse Gaussian distribution, while Iliou et al. (2017) and Yu et al. (2015) proposed the use of a Zipf-like distribution. Besides the total number of hyperlinks to visit, Iliou et al. (2017) used a uniform distribution for the selection of the total number of web pages to follow from a specific web page. Concerning the time that visitors should spend on each web page, Yu et al. (2011), Yu et al. (2015), and Iliou et al. (2017) proposed the use of (double) Pareto distribution, while Oikonomou and Mirkovic (2009) proposed the use of the uniform distribution. For the selection of which pages to visit, Yu et al. (2011) selects the first page to visit using

the a Zipf-like distribution to calculate the probability of visiting each web page, assuming that they have all the web pages of the web server ranked based on their popularity.

As we can see, works that use statistical distributions to model a humanlike behaviour require an adequate amount of knowledge about the web servers of interest and their visitors. Having this information, we see that it is possible to simulate a humanlike behaviour. For example, Yu et al. (2015) assume that they can obtain information such as the ranking of the page popularity of the target web site, the total number of requests to the web server, and the total number of requests to specific web pages. As it is evident, this information is rarely accessible and additional actions are required. Additionally, we see that behaviours on different servers are being simulated better with different distributions. This means that such techniques are limited to specific visitors of specific websites.

Markov Models have also been used in literature alone or in combination with other techniques for the modeling of browsing behaviour. In such approaches, the researchers consider that evasive bots need to know only their current state to decide which pages to follow and how to remain undetected. Specifically, Xie and Yu (2009) proposed the use of Hidden Semi-Markov Model to model the request rate, viewing time, and request sequence of a human browsing behaviour. They have seen that these depend on the structure of a website and the way users access those web pages. Additionally, Awad and Khalil (2012) proposed the use of a technique that combines a modified Markov model and a classifier to predict the next web pages that visitors will follow. This technique aims to also alleviate the problem with the several different paths that the visitor can follow in a web page. By assuming that the order of the web pages to visit does not matter as well as by performing a “pruning” technique to limit those paths, they were able to predict the next web pages that a visitor will follow based on the currently visited web pages. Furthermore, Yu et al. (2015) combined a Markov based model with the statistical distributions that simulate the duration of each session, the browsing length, and the probability to select a specific web page, to model a humanlike behaviour. However, as discussed above, information about the web site and their visitors are required for that.

Finally, machine learning based techniques have also been proposed to allow the better modeling of a humanlike browsing behaviour. As discussed above, Awad and Khalil (2012) used a machine learning based classifier in combination with a Markov model based approach to modelling a humanlike behaviour. Additionally, Iliou et al. (2017) calculated the contextual similarity of the new page to visit with the current one so as to make bots following web pages of similar topics to better simulate a humanlike behaviour. Ad-

ditionally, differentiating from the other works, in Iliou et al. (2017) the machine learning approach does not depend on the structure of the web server or the visitors browsing behaviour on the specific web server, since it is a more generic one.

As we can see, all the aforementioned approaches that model a humanlike behaviour depend (to some extent) to the unique characteristics of the specific target web servers as well as the behaviour of their visitors. Only the machine learning based text classifier proposed by Iliou et al. (2017) does not require such knowledge, but models a very small part of a humanlike behaviour. Thus, such approaches require this additional knowledge for the effective modeling of a humanlike browsing behaviour.

Evasion of Behaviour Based Detection Techniques that Use Mouse Trajectories

In the recent years, as web bot detection techniques started examining the mouse trajectories of visitors, web bots enhanced their evasive behaviour by mimicking humanlike mouse movements while browsing. However, there are only a few works proposed in literature for that. Specifically, current works in literature focus on (a) generating mobile touchscreen events, and (b) generating mouse movements to bypass specific challenges. For those, (i) heuristics, (ii) statistical distributions, (iii) parametric curves, and (iv) machine learning have been used to achieve that.

Chu et al. (2018) used heuristics to select random values to simulate various effects when bots perform mouse movements. In a similar manner, Wei et al. (2019) used heuristics when selecting the starting and ending point of a mouse trajectory to be simulated by the web bots, the “step” size of the mouse trajectories (i.e., the distance between each point/pixel that the mouse, when controlled by the web bot, hovers over), and the time interval (i.e. time difference) between two consecutive points. Additionally, Chu et al. (2018) proposed a bot that uses as input human mouse trajectories and repeats them (without exhibiting any intelligence). Furthermore, Acien et al. (2020b 2021) proposed the use of heuristics to model the velocity of mobile mouse touchscreen trajectories, by spacing the points of the trajectory on a log scale (emulating a velocity profile with acceleration similar to the one observed in human samples). Similar to the heuristic approaches used for the simulation of the browsing behaviour of humans, such techniques require knowledge about the specific web pages as well as the behaviours of visitors, which is not always easy to obtain, limiting their effectiveness.

Additionally, statistical distributions have been used to generate evasive behaviours. Similarly to the heuristic approaches, Wei et al. (2019) tested the use of uniform and

Gaussian distributions to simulate the “step” size of the mouse trajectories, and the time interval between two consecutive points. Furthermore, Acien et al. (2021) used statistical distributions to generate mobile touchscreen events. Specifically, they extracted several aspects from the mobile touchscreen trajectories of humans and noticed that all these follow the Gaussian distribution, which they used thereafter to generate additional values for those aspects that simulate a humanlike behaviour. The aspects that they examined were duration, distance, angle, mean velocity, move efficiency, and displacement. Again, these method requires knowledge of the behaviour of humans. Also, Acien et al. (2021) generated only the extracted information of the swipe trajectories, not the pixels with the respective timestamps that make the trajectory.

Parametric curves have been used by Wei et al. (2019) to generate humanlike mouse movements. Specifically, they used the Bézier curve to generate a smooth continuous curve that can look humanlike. Even though such curves present more humanlike mouse movements compared to simple straight lines, these can be easily distinguished visually from human ones.

Concerning the machine learning based techniques, Akrouf et al. (2019) proposed the use of Reinforcement Learning (RL) to bypass Google reCAPTCHA v3 by generating humanlike mouse movements. However, this technique only targeted to evade a specific CAPTCHA challenge instead of generating general humanlike mouse trajectories. Additionally, Acien et al. (2021) proposed the use of GANs with Long Short Term Memory (LSTM) network layers to generate synthetic swipe data that can be used by mobile web bots. These techniques have shown very promising results, as the RL approach works without any knowledge of the human behaviours, and the GANs are able to generate humanlike mobile trajectories.

2.4 Conclusions, Insights, and Limitations

In this chapter we presented the literature review on the web bot detection/evasion domain, starting from the web bot landscape, and then presenting the different signature-based and behaviour-based web bot detection and detection evasion techniques proposed in literature.

As discussed above, web bots vary in sophistication, with advanced web bots being especially appealing to malicious web bot creators, due to their browserlike fingerprint and humanlike behaviour which reduce their detectability. Such evasive web bots can

be and have been abused for malicious purposes, since (i) they can perform highly complex tasks, and (ii) avoid detection by presenting a browser fingerprint and a humanlike behaviour. This makes them particularly dangerous since they present themselves as humans and perform several actions in a humanlike way, which severely hinders their detectability.

Additionally, even though for many years the most popular techniques for detecting web bots were based on CAPTCHAs, current advances in machine learning that can be used by web bots have considerably reduced their effectiveness. Thus, current web bot detection approaches used by commercial solutions, combine (i) rule-based web bot detection based on browser fingerprinting techniques, with (ii) web bot detection based on the behaviour of the visitors.

However, we show that literature on behaviour-based web bot detection techniques does not focus on detecting only advanced web bots, but proposes general detection frameworks that target all web bots regardless of their sophistication level. Thus, even though most techniques proposed in literature achieve very high performance, these techniques might not be very effective when detecting only advanced web bots. This raises the question concerning the effectiveness of the state-of-the-art web bot detection techniques against (only) advanced web bots that try to evade detection, and the potential need of novel web bot detection techniques that target those advanced web bots (if current approaches achieve low performance). Such web bots should be considered particularly dangerous, as discussed in Section 2.1.

Moreover, we show that in the last years literature started focusing on using mouse movements for the detection of web bots, since it is harder for web bots to generate humanlike mouse movements compared to web logs. Considering the importance in detecting advanced web bots, there is a need for evaluating such detection techniques alone or in combination with more traditional detection techniques that use web logs against advanced web bots.

Also, we show that there are several types of evasive web bots proposed in literature, which use either a single technique or a combination of different techniques, including heuristic approaches, statistical distributions, statistical models, parametric curves, and machine learning. The latter (i.e., machine learning based techniques) are very promising since they usually do not depend on specific configurations and structure of a target web site. However, very few works in literature have used machine learning based techniques. This raises the question of how well the (state-of-the-art) web bot detection techniques

perform when faced with advanced web bots that use recent advances in machine learning to evade detection. As we discussed before, it is of utmost importance to detect such advanced web bots, since they can be very dangerous by presenting themselves as humans and performing similar actions as humans do.

3 Behaviour Based Web Bot Detection

As discussed in Chapter 2, the limitations of signature-based detection techniques, and the advances in browsing automation technologies that enabled the generation of browser-like fingerprints by web bots resulted in bot detection techniques to focus on visitors' behaviour. Well-known security companies' reports, such as Akamai (2021), Cloudflare (2021), DataDome (2021a), PerimeterX (2021), and Distil Networks (2019), showcase that these companies use and combine in their latest products signature-based detection techniques (primarily browser fingerprinting techniques) with machine learning methods that examine the visitors' behaviour to detect web bots.

However, as shown in Figure 2.2, only in recent years web bot detection frameworks have started being evaluated against advanced web bots. As discussed in Chapter 2, advanced web bots are particularly dangerous since they present themselves as humans and perform several actions in a humanlike way, which severely hinders their detectability. If malicious authors invest the time in creating advanced web bots, such web bots will be used for high impact attacks. Thus, web servers should consider creating detection frameworks that focus on advanced web bots. Additionally, later works from literature have been evaluated on both moderate and advanced web bots, instead of only advanced web bots, thus their true performance against only advanced web bots cannot be determined.

To this end, in this chapter we initially evaluate the web bot detection techniques that were state-of-the-art in the beginning of our research (i.e., before 2018) against web bots of different sophistication levels and examine how this affects the detection performance (Section 3.1). We showcase the limitations of the aforementioned web bot detection techniques in detecting advanced web bots that try to evade detection, a work that was published in Iliou et al. (2019).

Additionally, we propose a novel detection framework that can be used for the detection of advanced web bots (Section 3.2). Instead of trying to further improve traditional web detection techniques that use web logs for the detection of web bots (where sev-

eral works in literature have been proposed on that), we propose a novel approach that utilises the mouse movements of visitors.

We opted to do that since humanlike mouse movements are far more difficult to simulate compared to web browsing behaviours. Our approach was in line with other researchers, where they also started utilising mouse movements (in parallel with us, but without being aware of each others works before). Additionally, differentiating of all state-of-the-art approaches proposed in literature, the proposed framework comprises and advances two detection modules, one that uses web logs and one that uses mouse trajectories. The modules are used in a novel way to capture the different temporal and spatial characteristics of the types of data utilised, which results in a more robust detection framework able to detect web bots even in cases where one of the individual detection modules fails. This novel framework was published in Iliou et al. (2021a).

Next, we present in detail the detection framework used for the evaluation of the state-of-the-art web bot detection technique that use web logs (Section 3.1), followed by the novel web bot detection framework that combines web logs and mouse movements for the detection of advanced web bots (Section 3.2).

3.1 Machine Learning Based Detection Using Web Logs

As discussed before, in the beginning of our research (i.e., before 2018), literature focused on detecting web bots based on their behaviour using web logs. However, such approaches have not been evaluated against advanced web bots that try to evade detection, even though such bots could probably have been used for high impact attacks.

Thus, in this section we propose and evaluate a framework that uses the most prominent detection techniques at that time (i.e., before 2018) against web bots of different sophistication levels. The evaluation shows that even though such techniques are very effective against simple web bots (something that is in line with literature, as shown in the Table B.1 which summarises the performances of literature), such techniques fail to detect advanced web bots. This work was published in Iliou et al. (2019).

Next, we present the web bot detection framework (Section 3.1.1), followed by the evaluation setup (Section 3.1.2), and the evaluation results (Section 3.1.3) against simple and advanced web bots.

3.1.1 Web Bot Detection Framework

The web bot detection framework is based on and combines the most prevalent web bot detection techniques that had been proposed in literature when this research took place (i.e., before 2018). These techniques extract several measurable values (i.e., features) from the web logs and used them as input to train machine learning models for the detection of web bots.

The architecture of the web bot detection framework is shown in Figure 3.1. The input of the framework is a directory path in which the HTTP logs from the web server are stored. The framework uses a regular expression to extract the relevant content from HTTP logs. Thus, the process of applying different log files as input is trivial, since any new format of interest can be incorporated by only adapting this regular expression rule.

After the successful connection of the framework with the HTTP server log files, the session extraction procedure takes place, where HTTP log data are split into sessions. As discussed in Section 2.2.2, it is common for web bot detection frameworks to classify web bots and human visitors based on their total behaviour over a session, since only single requests do not include enough information for behaviour of visitors. For each session, a feature vector is created using a set of features proposed in literature. This feature vector aims to represent the general behaviour of the visitors throughout the session, which can be used to uncover the different characteristics of the behaviour of web bots and humans. After that, each session is annotated as web bot or human using an automated way. Furthermore, the importance and effectiveness of each feature is evaluated and a subset is selected, which is commonly used in machine learning. Finally, the selected feature vectors are used to create the classification models which are used for the classification of new sessions as bots or humans.

In the testing phase (i.e., when the detection framework is deployed on a running web server), the previously created classification models are used to identify web bot sessions in new unseen data. When new data are available, their sessions and features are extracted accordingly. Each classifier uses a unique subset of the available features which consists of the ones that were deemed most important during their training stage. The trained classifiers take the new data as input and determine whether each visitor is a bot or a human.

Next, we present the technical details of the proposed framework.

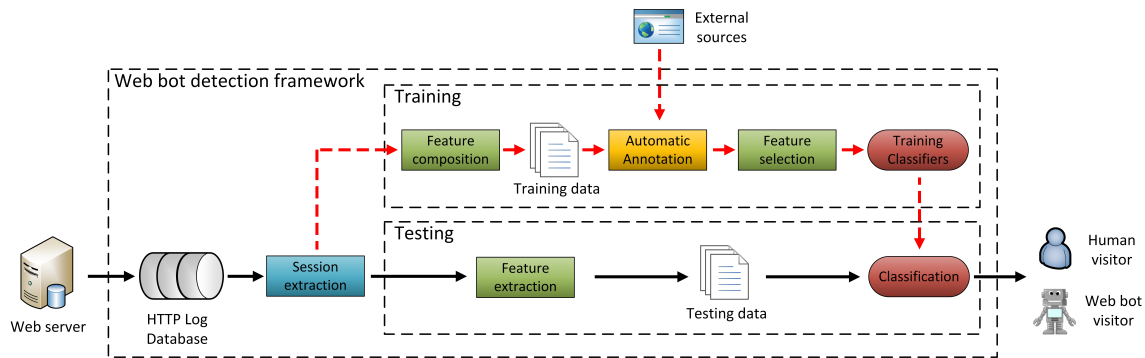


Figure 3.1: The web bot detection framework

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. As discussed in Chapter 2, it is common to use a combination of the IP with the browser agent name for the creation of a unique identifier per visitor, when no additional information about the visitors session exists (which is the default configuration of common web servers). Examples of works that follow this approach are Stevanovic et al. (2012 2013), Bhargav and Bhargav (2014), Zabihi et al. (2014), Grzanic et al. (2015), Rude and Doran (2015), Sisodia et al. (2015), Doran and Gokhale (2016), Hamidzadeh et al. (2018), Rovetta et al. (2017 2020), and Suchacka et al. (2021).

The aforementioned technique will not necessarily result in distinguishing all users from each other, since there might be two users with the same IP and agent name or one user changing several agent names in rotating order. To this end, more advanced fingerprinting techniques could have been used, such as the ones proposed by Nikiforakis et al. (2013), Azad et al. (2020), Laperdrix et al. (2020), and Schwarz et al. (2019), that extract information about browser-based characteristics (e.g. ActiveX support, Flash enabled, language enumeration, etc.), OS and applications features (e.g. OS and kernel version, Windows registry, etc.) and hardware features (e.g. screen resolution). However, since this information is not available in the web log data that we used, we followed the default approach of identifying separate sessions by the combination of a unique IP - agent name pair.

To define when a user session has ended, we use a 30 minute threshold, as proposed in Stevanovic et al. (2012 2013), Sisodia et al. (2015), Bhargav and Bhargav (2014), Zabihi et al. (2014), Rude and Doran (2015), Doran and Gokhale (2016), Hamidzadeh

et al. (2018), Rovetta et al. (2017 2020) and Suchacka et al. (2021). More specifically, when a session stays idle for more than 30 minutes (i.e., no request is performed from a specific IP - agent name pair for more than 30 minutes), a new session is created upon a new request. Furthermore, sessions that have a total number of HTTP requests lower than a threshold k are not taken into consideration because it is not feasible to distinguish bots and humans based on only a few HTTP requests, as shown by Rude and Doran (2015).

Feature Extraction

The information included in each session is encoded into measurable values and used as input to train the classification models. To decide which measurable values (i.e., features) to consider, we accumulated the most promising features that had been proposed over the past six years in the beginning of this research (i.e., 2012~2017). The total features proposed in literature are presented in Appendix A. Out of those, we selected 23 that were available/applicable to our environment, which are presented in Appendix C. For example, features specific for e-commerce sites, or features that required additional information that we currently did not have were not included. In short, to distinguish web bots from humans we considered the total requests that visitors make, the file types accessed, the values of the request headers, the URLs accessed and the respective navigational patterns, and the time between successive requests.

Automatic Annotation

The extracted sessions that are used for training the classifiers are annotated as “bot visitor sessions” or as “human visitor sessions”. Bot visitor sessions contain two different types of sessions; (i) those in which the web bots are conspicuous, i.e. they are not trying to hide the fact that they are bots, and (ii) those in which the web bots are inconspicuous, i.e. they replace one or more of their normal bot characteristics with those of a human visitor to remain undetected.

The annotation process which we followed is depicted in Figure 3.2 and it is a two step process. The first step is to identify simple web bots by examining whether their agent name is similar to a browser one, while the second one is to identify, to the best of our ability, web bots that present a browser fingerprint and, in some cases, a humanlike behaviour by using an external honeypot. Initially, we used the API provided

by Useragentstring¹, a server that classifies agent names in several categories such as “browser”, “crawler”, “library”, “link checker”, etc. After that, we used the API provided by GreyNoise², a server that collects and analyzes untargeted, widespread, and opportunistic scans and attacks or malicious activities, to check whether the IPs have been found to perform any of the above.

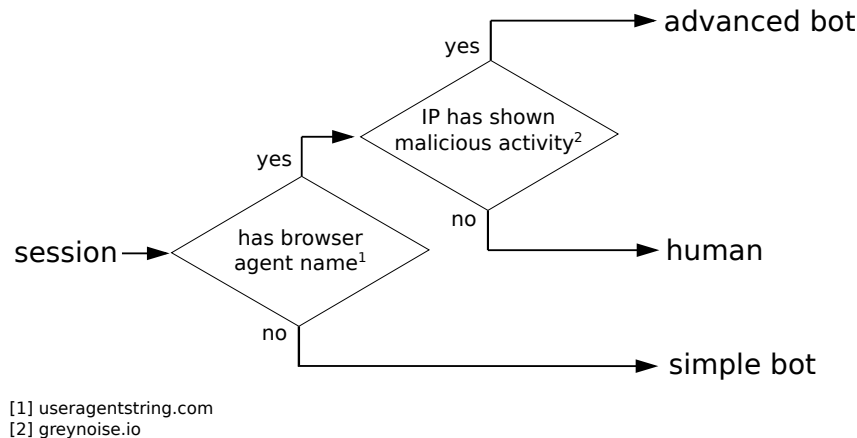


Figure 3.2: Automatic annotation process

The main idea behind our approach is that it is not common for a human visitor to change the agent name of their browser. Thus, if a session has a non-browser agent name, it can be safely annotated as web bot. However, as also mentioned by Distil Networks (2018), Jacob et al. (2012), and Yang et al. (2015), all sessions that have a browser agent name are not necessarily from human visitors, since bots might change their agent name to present a browser-like fingerprint.

Feature Analysis and Selection

Feature selection is described as a method whereby specific features are selected from the set of all available features. In machine learning, and more specifically, classification problems, some features might result in the decrease of the models’ accuracy, thus feature selection usually is performed to select the most promising features. However, feature selection is not very commonly used in the web bot detection domain. Specifically, Haidar and Elbassuoni (2017) and Suchacka et al. (2021) performed a feature selection step which was done primarily manually via several experiments or by checking the performance of features in other works. Zabihi et al. (2014) used a significance test

¹<http://useragentstring.com/>

²<https://greynoise.io/>

to see which features are significantly different between the bot and humans sessions. Finally, Hamidzadeh et al. (2018) and Zabihimayvan et al. (2017) proposed the use of FRS for the dynamic selection of the most promising features.

Furthermore, as discussed in Hamidzadeh et al. (2018), the best performing features can differ between different datasets. For example, they show that in one dataset, the feature that calculates the HTML-to-image ratio of the session requests is a very good separator, while in other datasets the percentage of 4xx requests found to be better, and in a third one the percentage of the 3xx requests was better.

Thus, our framework also supports the analysis of all the available features and the selection of the most important ones. Two selection modes are supported, one which selects the most promising features based on the feature analysis regardless of the classification algorithm employed and one that is classifier dependent. In both cases, each classifier is accompanied by a boolean array that indicates which features are to be used from all the available features for its training. The same features are used in the testing process.

Classification

The classification process consists of two phases, the training phase and the testing phase. In the training phase we feed the framework with known web bot and human sessions as input to create the classification models. These models are stored into the framework so that they can be used for the testing process.

The testing process is similar to the training one. The new (unseen by the classifiers) web sessions are used as input to the framework and the classifiers generate the respective label (web bot or human). To evaluate the framework, the labels of the sessions used for testing are known (but kept secret). However, in a real case scenario, the nature of these sessions would be unknown.

3.1.2 Evaluation

To assess the effectiveness of the framework that uses web logs for the detection of web bot sessions, a series of experiments were conducted using real HTTP traffic collected from a public web server. This section describes the evaluation methodology, the dataset, the feature analysis and selection process, the evaluation metrics considered and, finally, the classification algorithms tested and their configuration.

Evaluation Methodology

The purpose of this framework is to identify and show the unique challenges that arise when state-of-the-art web bot detection techniques are utilised for detecting advanced web bots as opposed to simple bots. To this end, we evaluated our framework on how well it can identify simple and advanced web bots separately. Initially, we identified the most important features in the case of simple and advanced Web bots. We used these features to generate the respective classification models and evaluate our framework after the models' deployment.

Furthermore, we took into account the fact that, in a real world case scenario, it is imperative to have a low false positive rate in order to avoid miscategorising human visitors. Thus, we tested our framework's general performance in various working points (i.e., classification thresholds) and analysed its performance on the working point in which the false positive rate is relatively low.

Dataset

To evaluate our work we created a dataset from our lab's web server. We opted to do that since there are no public datasets available on this domain and we were not able to get access to datasets from other works in literature. Specifically, the framework was tested on HTTP log data collected from MKLab's public web server.³ Instead of feeding the framework with data real time, we used a year's worth of HTTP log data (from 20/3/2016 to 20/3/2017) as input in a simulated time mode. Similarly to Rude and Doran (2015), we only considered web sessions with more than $k=30$ requests per session to ensure that the framework has adequate data to identify the nature of each visitor. The value of k was chosen heuristically.

We annotated the dataset by examining the agent name of the visitor as well as whether its IP has shown malicious activity (see Section 3.1.1). The total unique agent names extracted from sessions with more than $k=30$ requests were 2793. From them, the 2723 were annotated by the useragentstring's API as "browser" (2628) or "bot" (95) and 70 were annotated as "unknown". For the "unknown" agent names, we manually annotated them as browsers (66) or bots (4).

As we mentioned in Section 3.1.1, the IPs of the sessions that were annotated as "browsers" by the useragentstring's API (15452) are also checked for malicious activity

³Multimedia Knowledge and Social Media Analytics Laboratory, <https://mklab.itl.gr/>

using the GreyNoise’s API. Thus, we end up changing the annotation of 299 unique IPs (554 sessions) which were originally annotated as “browser” but their IPs have been marked as bots by GreyNoise. The total unique agent names and IPs per class (i.e. browser, simple bot, advanced bot) are shown in Table 3.1.

Table 3.1: Unique agent names and IPs

	Bots			Humans	Total
	<i>Simple</i>	<i>Advanced</i>	<i>Total</i>		
Agent names	99	105	204	2589	2793
IPs	602	299	901	15153	16054

To evaluate the framework, we split the dataset into two sets, one for training and one for testing. Our web server by default splits the HTTP log data into files based on a log rotation technique.⁴ The total files that were generated over a year were 13. We grouped the files into two packages, (i) the training one using the first 8 files (from 20/3/2016 to 4/12/2016) and (ii) the testing one using the other 3 files (from 4/12/2016 to 20/3/2017). For each of these files we extracted all sessions with more than $k=30$ requests per session, following a similar approach to Rude and Doran (2015).

The final number of extracted sessions is shown in Table 3.2. To assess the framework’s performance in identifying simple bots and advanced bots separately, we created two “sub-datasets”, the D1 that contains the human and simple bot sessions and the D2 that contains the human and the advanced web bot sessions.

Table 3.2: Human and Web bot sessions

	Bots			Humans	Total	Total
	<i>Simple</i>	<i>Advanced</i>	<i>Total</i>		D1	D2
Train	1321	431	1752	17462	18783	17893
Test	1034	123	1157	6195	7229	6318
Total	2355	554	2909	23657	26012	24211

By comparing our dataset with the datasets used in literature (based on the information that is included in the respective papers, since these datasets were not publicly available), we see that the total traffic of our web server was far less than in the ones from the literature. For example, Cabri et al. (2018) and Suchacka et al. (2021) observed half

⁴<https://httpd.apache.org/docs/2.4/logs.html>

the amount that we collected (which was over period of one year, as discussed above) in only one month, while Stevanovic et al. (2013) observed more than double the amount of our data in one month. However, those works only considered one month of log data, instead of a whole year, so our dataset (in total size) was comparable. Also, having a dataset that spans over one year can be considered as an advantage, since web bots' behaviour can change over a specific period during the year (e.g., summer period vs rest of the year). Thus our evaluation covers these cases as well.

Additionally, we see that most of our sessions were annotated as humans (about ~ 89), which is contradictory to the percentages presented in Distil Networks (2017) and Distil Networks (2018), two reports of Distil Networks (now acquired by Imperva), a lead in web bot detection, for the respective periods of time. Specifically, in Distil Networks (2017) and Distil Networks (2018) it is mentioned that in 2016 and 2017 61.3% and 57.8% of the total traffic was found to be from human visitors (respectively). Concerning the datasets used in literature, the most recent one collected in 2018 from Almahmoud et al. (2019), where the authors observed that the simple bots were only 1.22% of the users (while there is no other information about the total human and bot sessions). Older datasets, such as datasets collected in Doran and Gokhale (2016) and Zabihimayvan et al. (2017) collected in 2009 and 2013 respectively (in these works a few datasets were collected), had a relatively low percentage of human sessions. Specifically, they observed the human sessions to be 14.5% and 38.5% of the total sessions respectively. However, similar works, such as Sisodia et al. (2015), Hamidzadeh et al. (2018), and Loyola-González et al. (2018), (not mentioning the year the datasets were collected) observed a percentage of human sessions of 93.1%, 93.3%, and 97.8% (respectively), which were closer to our dataset.

Furthermore, we see that the percentage of simple vs advanced web bots we observed in our dataset is close to Distil Networks (2017) and Distil Networks (2018). Specifically, in our dataset, $\sim 19\%$ of the web bots were advanced which is similar to the 19.9% and 21.2% reported in Distil Networks (2017) and Distil Networks (2018) (respectively). Concerning the equivalent percentages of works proposed in literature, there was no distinction between simple and advanced web bots, thus we cannot compare our dataset with those on that aspect.

Thus, we see that there are some commonalities of our dataset with the ones proposed in literature and web bot detection companies, in regards to the percentages of the bots of different sophistication levels and humans. Additionally, we see that there

are differences in the percentages and volume of data even among different works in literature. This can be attributed to the fact that (i) the amount of the bots targeting a specific web server depends on many things, including the content hosted on the server as well as their services, and (ii) there are differences in the bot and human annotation process in literature making the percentage of human sessions dependant on the annotation process.

Feature Analysis and Selection

To analyse the importance of the extracted features in the detection of simple as well as advanced web bots we utilised the Principal Component Analysis (PCA) and the χ^2 (chi-square) feature selection techniques. For both of these techniques the data were scaled. When scaling the data for the PCA, we subtracted the mean values and then divided by standard deviation for each feature in the training set. In the case of χ^2 , we divided by standard deviation without subtracting the mean to avoid negative values. We then used the mean and standard deviation values calculated from the training set to perform the same process in the testing set.

PCA can be used to assess the importance of each feature by calculating its contribution to the generated principal components. To do that, we followed the work of Johannes (2016) and measured the mean of each feature “contribution” to all the generated components of the PCA using the training set (D1 for simple bots and D2 for advanced bots). Usually, as Johannes (2016) mentions, the smaller principal components (i.e., with lower variance) are associated with noise and thus they can be omitted. Thus, the features with the lowest cumulative “contribution” to all the principal components can also be associated with noise. However, the high variance principal components are not necessarily all useful, since they might not be correlated with the respective class (i.e., web bot or human) or they may refer to noise existing within the data. Thus, we combined the results of the PCA technique with the χ^2 feature selection technique to see the most important features to the web bot detection problem.

To select the features that will give us the highest score for each classifier, we used the greedy Sequential Feature Selection (SFS) technique presented in Pudil et al. (1994). The SFS works as a wrapper on top of each classifier. It is an iterative process where in each iteration the feature with the highest metric (in our case balanced accuracy) on the training set is chosen and added to the features that are used for each classifier. By this way, the features that perform the worst will be added in the end and can be omitted if

they do not contribute to the performance. Thus, SFS provides different results for each classifier, which is useful because each feature may contribute differently depending on the classifier that was used.

Evaluation Metrics

Several researchers used accuracy as the evaluation metric for web bot detection, such as Rude and Doran (2015), Stevanovic et al. (2012), Wang et al. (2013), and Rovetta et al. (2017). Since it is possible for an algorithm to have high accuracy while maintaining low precision, Alam et al. (2014), Wang et al. (2015a), Doran and Gokhale (2016), Sisodia et al. (2015), Stevanovic et al. (2012), and Rovetta et al. (2017) use the precision and recall metrics as well to assess the performance of the proposed approaches more accurately. Furthermore, Wang et al. (2015a), Doran and Gokhale (2016), Sisodia et al. (2015), Stevanovic et al. (2012), and Rovetta et al. (2017) calculated the harmonic mean of the precision and the recall which is called F-measure, F1 score or simply F-score.

Due to the unbalanced classes in our dataset (something that is commonly observed in this domain, as shown in the dataset section), we decided to use balanced accuracy as opposed to accuracy. Furthermore, to evaluate the framework's performance in both the case of web bot detection as well as human user detection we calculated the precision, recall, and their harmonic mean, F-score, for both classes. Finally, to gain a more general understanding of the performance of the classifiers irrespective of the working point (i.e., classification threshold) we considered the Area Under Curve (AUC) evaluation metric, presented in Fawcett (2006), that can be calculated by plotting the Receiver Operating Characteristic (ROC) curve for a classifier.

Classification Algorithms Tested

Our framework is built to allow for the effortless incorporation of any machine learning algorithm. As discussed in Section 2.2.2 and shown in Appendix B, there is no clear performance benefit of one algorithm compared to the others and some algorithms might work well in some datasets while fail in others. Thus, for our experiments we selected the most popular classification algorithms in this domain (as most works prefer classification over clustering). Specifically, we incorporated (i) the SVM, (ii) the Random Forest, (iii) the Adaboost, and (iv) the MLP Classifier. Furthermore, following the work of Sisodia et al. (2015), we added an ensemble classifier, which we call the Voting classifier, that performs a class probability averaging of all the available classifiers. We did not use Naive

Bayes (even though being one of the most popular ones), because it achieved very low performance in some works such as Stevanovic et al. (2012) and Sisodia et al. (2015) to avoid having a similar behaviour in our dataset.

Please note that even though the specific algorithms might slightly affect the performance of our framework (as shown in Appendix B for other works in literature), the main aim of this research is to examine the performance difference when trying to detect simple vs advanced web bots. Thus, the specific algorithms used should not play a very important role for our purpose, as long as they achieve similar performances to the ones presented in literature.

The parameters for each classifier are shown in Table 3.3. We performed an exhaustive search over specified parameter values for each classifier and chose the ones which have the highest balanced accuracy using a 2-fold cross validation on the training data. Furthermore, in the case of SVM and MLP Classifier, the data were scaled to avoid the problem of domination of some features over the others. To scale the data, we followed the same scaling technique that we used in the PCA (Section 3.1.2).

For the implementation of these algorithms the scikit-learn⁵ Python library was used. Furthermore, all the experiments were performed on an Intel processor at 3.4GHz and 32GB RAM for loading large datasets during the experiments.

3.1.3 Results

In this section we present the results of the evaluation of our framework in regards to detecting simple and advanced web bots. First, we analyse and select the most important features for each classification algorithm. Subsequently, we evaluate the general performance of our framework, and the performance of our framework in a false positive intolerant server.

Feature Analysis and Selection

Feature Analysis: Figure 3.3 presents the cumulative variance of the data by adding the PCA's principal components one at a time ordered by descending eigenvalues for the simple (D1) as well as the advanced (D2) web bots. As we can see, the principal components generated from D1 and D2 have similar variance. Moreover, the number of principal components which are essential to maintain at least 90% and 95% of the variance for simple and advanced web bots is 14 and 16, respectively.

⁵<http://scikit-learn.org/stable/>

Table 3.3: The parameters used on the classification algorithms.

Classification Algorithm	D1 - Simple Web bots	D2 - Advanced Web bots
SVC	RBF kernel, C=16384, gamma = $1.22 * 10^{-4}$, tol=0.001	RBF kernel, C=64, gamma=2, tol=0.001
MLP Classifier	tanh activation, adam solver, a=0.1, b1=0.9, b2=0.9, e= 10^{-5} , hidden layer sizes: (100, 50), constant learning rate	tanh activation, sgd solver, a=1, b1=0.1, b2=0.1, e= 10^{-8} , hidden layer sizes: (400), invscaling learning rate
Random Forest	Estimators=200, Gini crit., max features= $\sqrt{no_features}$, min samples per leaf = 4, min samples split 10, max depth=70	Estimators = 1000, Gini crit., max features= $\sqrt{no_features}$, min samples per leaf = 4, min samples split 2, max depth=10, out-of-bag samples
Adaboost	Decision Tree Classifier as base estimator, estimators=450, decision entropy criterion, no max depth, max features = $\sqrt{no_features}$, “best” split strategy, learning rate=1	Decision Tree Classifier as base estimator, estimators=50, decision entropy criterion, no max depth, max features= $\sqrt{no_features}$, “best” split strategy, learning rate=1

Furthermore, we calculated the absolute value of the mean values of each feature “contribution” to all the components for the simple (D1) and advanced (D2) web bots (Figures 3.4).

In general, the higher the mean value of the features’ contribution to the principal components, the more important the feature can be considered. However, this is not always the case; some of these features might not contribute to the problem. Thus, to decide which features are the most important, we also calculated the respective x^2 scores for the simple (D1) as well as the advanced (D2) web bots (Table 3.4).

In both cases (PCA and x^2), the importance of the features differs when using the D1

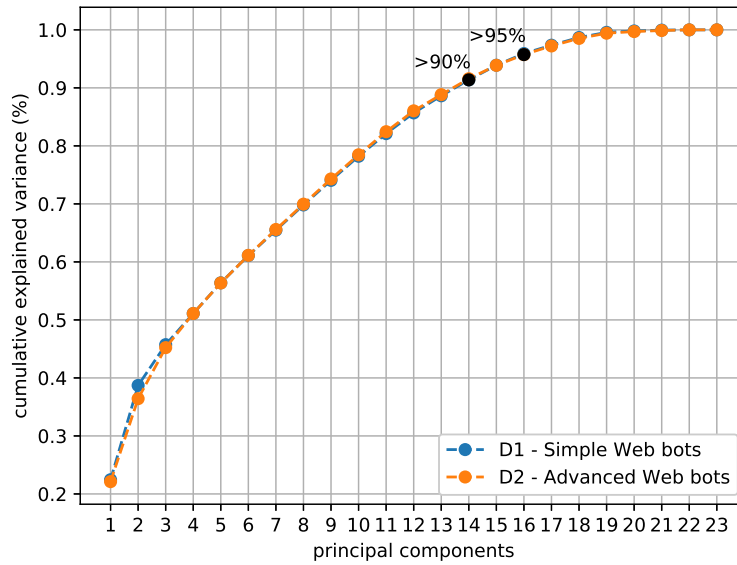


Figure 3.3: Cumulative variance of PCA's components for simple (D1) and advanced (D2) web bots

Table 3.4: Ranked features using x^2 for simple (D1) and advanced (D2) web bots

Dataset	Ranked features using x^2
D1 - Simple Web bots	13, 21, 23, 9, 15, 5, 10, 11, 8, 14, 7, 22, 16, 4, 6, 17, 12, 18, 19, 3, 20, 2, 1
D2 - Advanced Web bots	17, 4, 21, 12, 18, 1, 6, 3, 22, 23, 8, 9, 16, 11, 7, 13, 10, 5, 14, 2, 20, 15, 19

dataset (simple web bots) and when using the D2 (advanced web bots). Furthermore, the features that have both the highest contribution to the PCA and a high x^2 score are 21 and 23 for simple web bots (D1) and 17 and 18 for advanced web bots (D2). Features 21 and 23 have to do with time-related aspects of the browsing behaviour of the visitor. Simple web bots, usually, have a predefined and therefore predictable behaviour regarding time, which is why they can be detected this way. On the other hand, advanced web bots use a more unpredictable browsing behaviour, so the characteristic that gives them away is the unique content they try to access (features 17 and 18).

Feature Selection: Each feature might contribute differently in the performance of different classification algorithms. Thus, we performed the greedy SFS technique to pick the features that give the highest score (in our case balanced accuracy) for each classifier in the case of simple (D1) as well as advanced (D2) web bots. We decided to keep as many

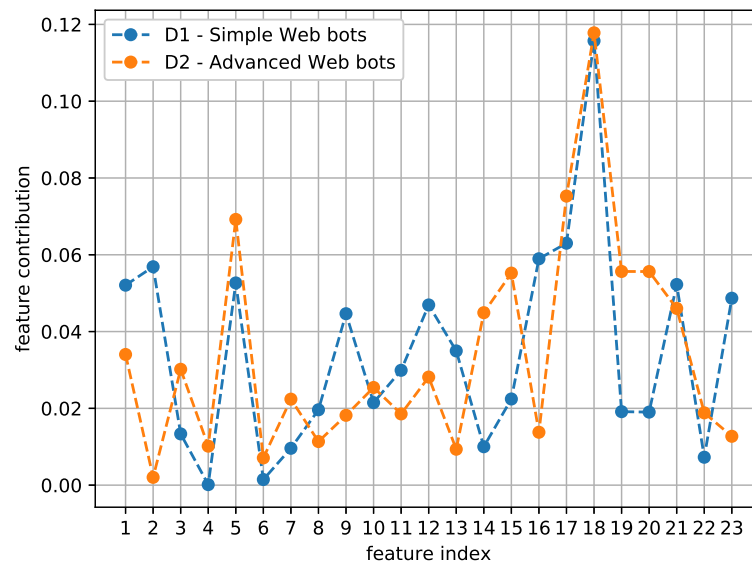


Figure 3.4: The absolute mean value of each feature contribution to PCA components for the simple (D1) and advanced (D2) web bots

features as possible as long as they do not noticeably decrease the balanced accuracy in training. The selected features for each classifier and dataset are shown in Table 3.5.

The SFS results show that each feature contributes differently in each classifier. Furthermore, the initial features selected by the PCA in combination with the x^2 were selected and highly ranked in some classifiers and rejected in other classifiers. Such an example is feature 17, which was initially selected and is highly ranked in the case of SVM and MLP Classifier, but rejected in the case of Random Forest and Adaboost.

For this reason, depending on the size of the dataset and the processing power we have, we can either select the most promising features according to the combination of the PCA with the x^2 technique or perform the greedy SFS over all the features and pick the ones that perform better on the training set. In our case, since the dataset was relatively small, we followed the latter.

Additionally, we see that the most promising features in our work may differ from the ones proposed in literature, something that is also discussed in Hamidzadeh et al. (2018). For example, for the simple web bots, we see that the best performing feature had to do with the requests with unsigned refer (feature 13). As discussed in Hamidzadeh et al. (2018), this feature is very popular in literature, but they found it not to be the most representative one in most of their datasets. This can be attributed to the fact that different

Table 3.5: Ranked and selected (bold and in brackets) features using SFS for simple (D1) and advanced (D2) web bots for each classification algorithm

Classification Algorithm	D1 - Simple Web bots	D2 - Advanced Web bots
SVM	{ 13, 4, 20, 8, 22, 14, 18, 21, 16, 1, 11, 19, 10, 17, 23, 5, 9, 12, 3, 6, 2, 7, 15 }	{ 17, 9, 20, 7 }, 5, 2, 14, 22, 6, 4, 11, 10, 8, 23, 3, 1, 16, 12, 19, 13, 15, 21, 18
MLP Classifier	{ 13, 15, 4, 14, 23, 20, 18, 5, 2, 21, 17, 12, 7, 6, 22, 9, 1, 8, 3, 10, 16, 11 }, 19	{ 17, 21, 23, 6, 12, 9, 19, 5, 22, 14, 18, 16, 20, 10, 1, 8, 11, 15, 2, 3, 7, 13, 4 }
Random Forest	{ 13, 4, 11, 19, 14, 20, 6, 17, 23, 5, 9, 12, 10, 10, 8, 21, 18, 1, 16 }, 22, 7, 2, 15, 3	{ 11, 14, 12, 20, 10, 5, 21, 7 }, 9, 6, 8, 4, 2, 17, 22, 23, 1, 15, 3, 16, 18, 19, 13
Adaboost	{ 13, 4, 19, 14, 20, 15, 5, 9, 10, 8, 11, 7, 17, 18, 12, 3, 6, 16, 1 }, 21, 23, 2, 22	{ 11, 8, 14, 20, 10, 5, 9 }, 2, 22, 7, 6, 4, 12, 1, 3, 17, 18, 21, 23, 19, 15, 13, 16

works in literature used datasets collected from different web servers. Each server having its own structure and content can attract different types of bots, therefore some features are more useful than others to separate web bots from human visitors.

General Performance

To evaluate the general performance of our framework, we plotted the ROC curve of the Voting classifier when the framework was tested on simple and advanced web bots (Figure 3.5). We also marked a few working points (i.e., classification thresholds) based on the respective False Positive Rate (FPR). We opted to do this because in a real-world scenario a web bot detection framework must be false-positive intolerant to avoid affecting the visitors' experience, even though usually literature does not consider that, and uses a default threshold of 0.5.

The performance of our classifiers shows that detecting simple web bots is a trivial task. The framework is able to effectively detect the simple web bots (D1 dataset) with an AUC=1.00. This is in line with the results of the literature, as shown in Appendix B. However, detecting advanced web bots (D2 dataset) is not that simple. The framework performs poorly and, if a low FPR is required, the framework detects very few web bots. This

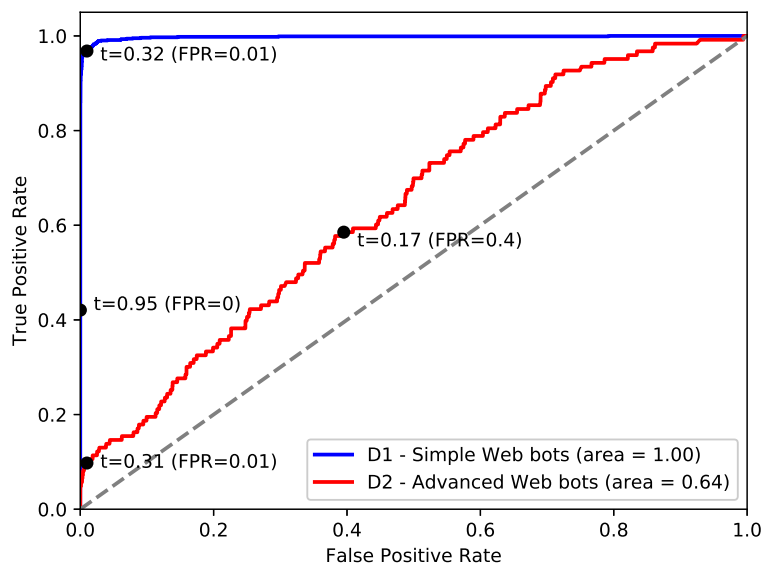


Figure 3.5: ROC curve of the Voting classifier for the simple (D1) and advanced (D2) web bots

outlines the fact that, even though the techniques proposed in literature for the detection of web bots based on the web logs that they generate achieved very high performance in detecting simple web bots, their performance considerably decreases when detecting advanced web bots, making them ineffective.

To further analyse the behaviour of our framework on the selected working points in the case of advanced web bots (D2), we calculated the confusion matrix of the Voting classifier on the two working points selected in Figure 3.5 (Table 3.6 and Table 3.7).

Table 3.6: Confusion matrix for advanced web bots (FPR=0.4, $t=0.17$)

		Predicted Values		Total
		Bot	Human	
Actual Values	Bot	82	41	123
	Human	3661	2534	6195
Total		3743	2575	

The choice of a working point depends on how strict we want our detection framework to be in each case. For example, choosing a working point with FPR=0.4, we would correctly identify 2 out of 3 advanced web bots, but most humans would be misclassified (Table 3.6). Choosing a higher threshold (lower FPR) results in fewer misclassified human

Table 3.7: Confusion matrix for advanced web bots (FPR=0.01, t=0.31)

		Predicted Values		Total
		Bot	Human	
Actual Values	Bot	18	105	123
	Human	417	5778	6195
Total		435	5883	

visitors, but the framework's effectiveness in detecting advanced web bots is decreased.

Performance on a False Positive Intolerant Web Server

To assess the framework's performance on a false-positive intolerant web server, we calculated the precision, recall, and F-measure in the working point of FPR=0.01 for all employed classifiers. Furthermore, we calculated the balanced accuracy, which represents more accurately the performance of the framework in the case of unbalanced datasets. The results are shown in Figure 3.6.

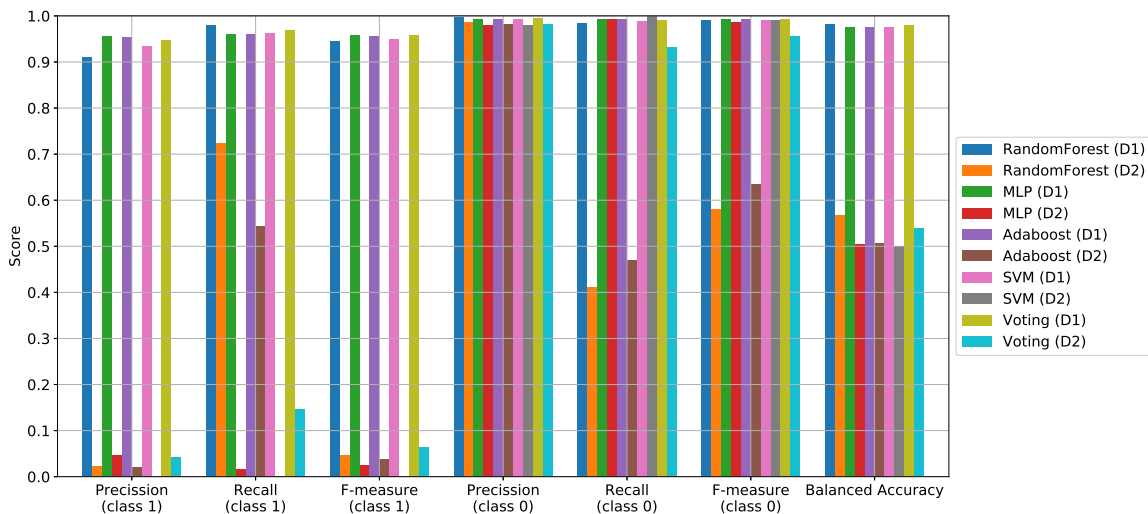


Figure 3.6: Comparison of the effectiveness of the classification algorithms for humans (class 0) and web bots (class 1) for the D1 (simple web bots) and D2 (advanced web bots) datasets in the working point with FPR=0.01

The performance of the classifiers shows that identifying advanced web bots is more challenging than identifying simple web bots. When choosing a working point with a low FPR, simple bots are detected with very high precision ($\sim 95\%$) and recall ($\sim 97\%$) which makes an F-measure higher than 96%. Furthermore, in the case of detecting human visitors (class 0), we achieved a precision and recall of more than 99% each. This comes

in line with the performance of literature, where, as presented in Appendix B, most works that detect simple web bots (both the ones that use classification and clustering) achieve a performance of more than 90% to several evaluation metrics, with some of them such as Rovetta et al. (2020) achieving F-measure and accuracy of more than 98.5%.

On the other hand, the framework achieves low precision and recall in the case of advanced web bots which results in a low balanced accuracy ($\sim 55\%$). Since current literature approaches that detect web bots based on their web logs have not been tested against advanced web bots, we cannot compare our results with what other researchers have observed.

Moreover, we see that the performance of different classifiers varies, something that is also common in literature (see Appendix B). To achieve a more balanced behaviour we chose the Voting classifier to be the main classifier. Generally, voting classifiers are not always guaranteed to have a better performance. However, they can be more “stable”, since, if one of the employed classifiers underperforms, its behaviour will be masked by the other classifiers. For example, Random Forest achieves the highest balanced accuracy in the case of advanced web bots (D2) but, at the same time, very low recall of the human class (see Figure 3.6).

3.1.4 Discussion

As discussed in Chapter 2, there is a huge incentive for individuals and companies alike to create web bots that can bypass web bot detection techniques. This has led to the introduction of advanced web bots that try to evade detection. In the experiments conducted in Section 3.1, we used a dataset which comprises the logs from a public web server, containing several sessions from such bots. We used these logs to determine the effectiveness of state-of-the-art web bot detection techniques against advanced web bots.

The results have shown that, although detecting simple bots is relatively easy (something expected from the works proposed in literature as well), detecting advanced web bots that present a browser fingerprint and, in some cases, a humanlike behaviour is much more difficult. Furthermore, if we try to detect such bots with current detection techniques, we will end up misclassifying benign visitors, which is a non-desirable behaviour in a real-world case scenario.

Additionally, we show that the ineffectiveness of the state-of-the-art approaches proposed in literature at the beginning of this research (i.e., before 2018) was not evident,

because either advanced web bots were annotated as humans (based on the current annotation methods that examine visitors fingerprint) or literature focused on identifying all kinds of web bots, treating web bots of different sophistication level as one group of visitors. Concerning the latter, since advanced web bots can be considerably fewer than simple bots (as discussed in Section 3.1.2), the aforementioned technique will present biased results masking its ability or lack thereof to detect advanced bots.

This ineffectiveness of those approaches against advanced web bots indicate the need for additional features and techniques for the detection of advanced web bots. This is also evident in literature, where the researchers changed direction towards using the visitor mouse movements for the detection of more advanced web bots (as discussed in Section 2.2.2). Additionally, as discussed in Chapter 2, since creating advanced malicious web bots requires considerably more effort by malicious actors, such bots will probably be used for attacks with high Return on Investment (ROI), which makes them particularly dangerous.

Thus, there is a need to apply more advanced security measures to protect such services. Such techniques may consider additional features that can be extracted from web logs to better distinguish web bots from humans, or different algorithms that have not currently been tested on this domain (such as recent advances in machine learning that utilise deep neural networks), or additional input data can be considered to make the generation of a humanlike behaviour from web bots harder (such as the use of mouse trajectories). Since the detection of web bots using web logs was a relatively well studied area with several works being published on that, we opted to investigate the latter approach in the following section (i.e., using additional input data, and more specifically, mouse movements) for the detection of more advanced web bots, something that, as discussed in Section 2.2.2, has also been followed and published by other researchers (at the same time as us). This does not mean that the former two approaches (i.e., adding more features or using more recent deep learning techniques would not increase the effectiveness of the detection frameworks). For example, one could model the requests as a sequence and use deep learning architectures such as RNNs, something that has shown very promising results in different areas.

3.2 Machine Learning Based Detection by Combining Web Logs with Mouse Behavioural Biometrics

In the previous section we have shown that current state-of-the-art web bot detection approaches that utilise web logs are not very effective against advanced web logs. Even though additional techniques could be tested on detection using web logs (such as the use of additional features or more advanced deep learning techniques that), since this area is relatively well investigated, we decided to follow a novel approach of including additional input data for the detection of web bots, and, more specifically, the mouse movements that they perform. Mouse movements are a good candidate since it is much harder to generate behaviours that simulate humanlike mouse movements compared to web logs, as humanlike mouse movements can become very complex in specific web pages to simulate. Furthermore, since web logs can be effective in detecting web bots of specific behaviours, we decided to propose a novel method that fusions the two individual approaches (i.e. the detection using web logs, and the detection using mouse movements) to get the benefits of both approaches.

Specifically, this novel framework uses as a base the web bot detection framework that was presented in Section 3.1, and advances it by introducing an additional module that uses mouse trajectories for the detection of web bots. Furthermore, the proposed novel detection framework comprising the two detection modules in a unique way to capture the different temporal and spatial characteristics that they provide. This results in a more robust detection framework able to detect web bots even in cases where one of the individual detection modules fail. This work was published in Iliou et al. (2021a).

Next, in Section 3.2.1 we present the detection framework, and in Section 3.2.2 we detail the classification methods used by each of the detection modules. In Section 3.2.3 we outline the evaluation process of the framework, and in Section 3.2.4 we include the results of the evaluation.

3.2.1 Web Bot Detection Framework

As discussed before, the detection framework fusions two detection modules, one that uses web logs and one that uses mouse movements. Since the two input data have different temporal and spatial characteristics, the proposed framework fusions the two detection modules in a way to preserve those. Thus, instead of doing a feature based

fusion, each of the modules builds its own classifier.

The first module (i.e., the one that uses web logs) is based on the web bot detection framework presented in Section 3.1 and 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. The second module takes advantage of the difficulty in simulating humanlike mouse movements and utilises features that can be extracted from the visitors' mouse movements to detect web bots. For the latter, we process the mouse trajectories as images and use them as input to CNNs to take advantage of the exceptional performance of the CNNs in identifying patterns in images. This technique has also been followed by Wei et al. (2019), a research that was done and published in parallel with our work.

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 3.7. 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 means 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.

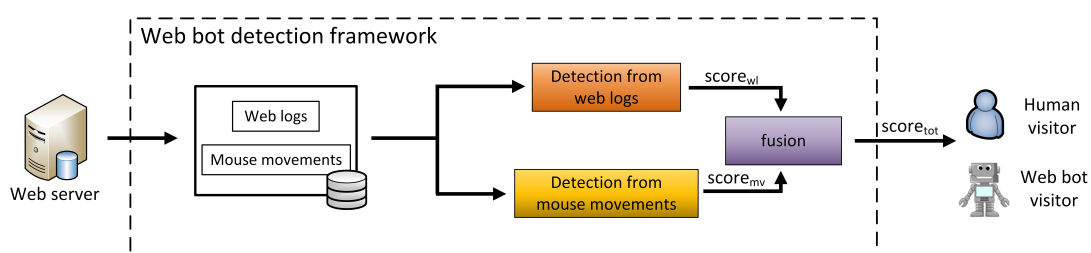


Figure 3.7: Web bot detection framework architecture

Next, we describe in detail the detection module that uses web logs, the detection module that uses mouse movements, and the fusion process.

Detection from Web Logs

The module that uses web logs to detect web bots is based on the one presented in Section 3.1 and is derived from the most prevalent relevant techniques that have been proposed in literature in the beginning of the research (i.e., before 2018). The architecture of this module is shown in Figure 3.8. 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.

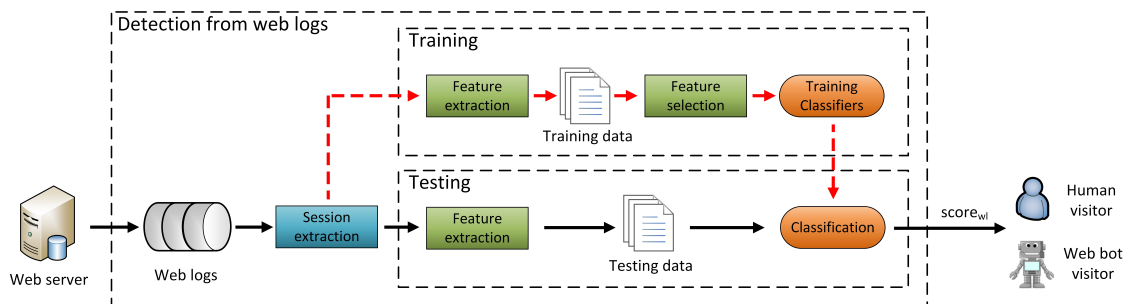


Figure 3.8: Web bot detection module that uses web logs

Similarly to Section 3.1, after the successful connection of the framework with the web server logs, the session extraction procedure takes place where the data are split into sessions based on some visitors' unique characteristics (to separate the data of different users) and based on a session timeout (to split the data of the same user into different sessions). For each session, several measurable properties/characteristics (features) of the visitors' behaviour are calculated. Subsequently, the most effective features are selected. The final feature vectors are used to create the classification models. 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. After that, the most effective features found in the training process are selected. These data are used as input to the (already) trained classifiers to label the new visitors as web bots or humans. Next, we present the details of the aforementioned process. We present these steps in detail below.

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. Differentiating from what we did in Section 3.1 which is the most prevalent approach in literature (as discussed

Section 3.1) where we combine the IP with the browser agent name for the creation of a unique identifier per visitor, in this section we use the PHP session ID along with the HTTP log data. As discussed in Section 2.2.2, the combination of IP with the browser agent name for user identification has flaws and is followed primarily when no additional information exists or is 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, similarly to Stevanovic et al. (2012 2013), Sisodia et al. (2015), Bhargav and Bhargav (2014), Zabihi et al. (2014), Rude and Doran (2015), Doran and Gokhale (2016), Hamidzadeh et al. (2018), Rovetta et al. (2017 2020) and Suchacka et al. (2021).

Feature Extraction: Similarly to Section 3.1.1, 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. The specific features used by our framework are presented in Section 3.2.2.

Feature Selection: As discussed in Section 3.1.1, in machine learning, some of the available features might negatively affect the effectiveness of the classification models. However, feature selection is not very commonly used in the web bot detection domain, with Haidar and Elbassuoni (2017) and Suchacka et al. (2021) performing feature selection manually based on performance, Zabihi et al. (2014) performed a significance test for each feature, and Hamidzadeh et al. (2018) and Zabihimayvan et al. (2017) used FRS for dynamic feature selection. The proposed 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 the evaluation section (Section 3.2.3).

Classification: The final feature vectors are used as input to train the classifiers. As discussed in Section 3.1.1, the framework supports the construction of an ensemble from several classifiers (i.e., it performs a class probability averaging of all the available classifiers). The classification algorithms selected were the same as the ones used in Section 3.1, and were selected based on their popularity and effectiveness. Details of those classifiers are presented in Section 3.2.2. In the testing phase, the ensemble is used to

identify whether new web sessions are from web bots or humans.

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, as discussed before, is very promising in detecting (advanced) web bots. The framework constantly collects the mouse movements of each visitor along with the respective timestamps. This information is then processed and used to train classification models for the detection of web bots based on their mouse movements. The architecture of this module is presented in Figure 3.9.

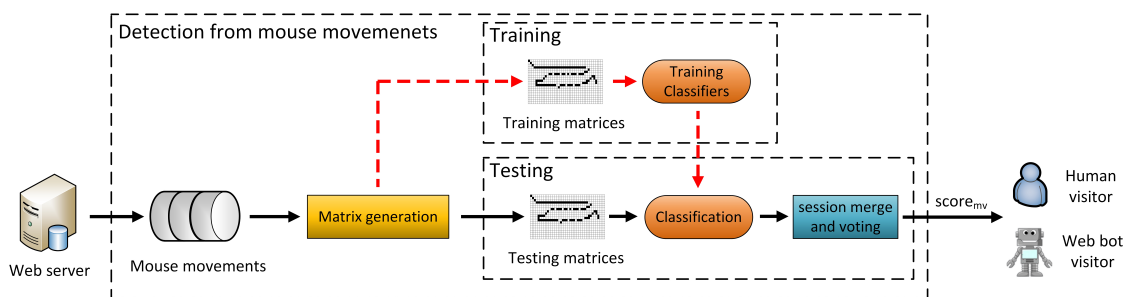


Figure 3.9: Web bot detection module that uses mouse movements

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 $\{(x_1, y_1, t_1), (x_2, y_2, t_2), \dots, (x_n, y_n, t_n)\}$, 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 3.10. To enable our framework to collect such data, a JavaScript file is embedded in each web page.⁶ This JavaScript file constantly locally stores 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

⁶Please 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.

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.

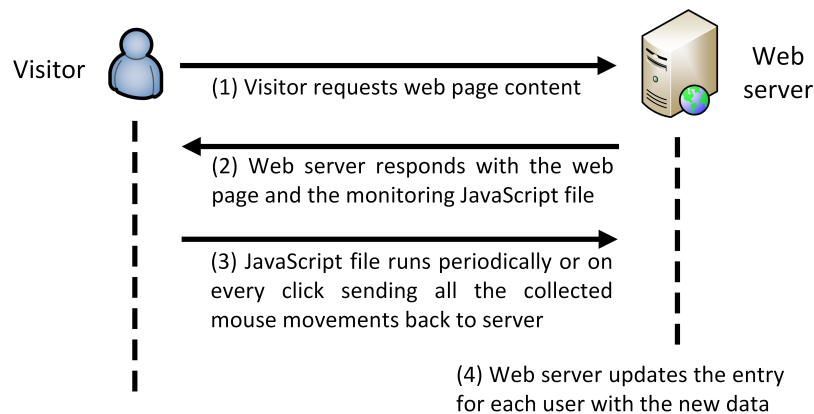


Figure 3.10: Mouse movement collection process

Classification: Since we aim to train our models to identify patterns in the visitors' mouse movements, 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 $\{(x_1, y_1, dt_1), (x_2, y_2, dt_2), \dots, (x_{n-1}, y_{n-1}, dt_{n-1})\}$, 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 $(x_n, y_n, 0)$.

Instead of extracting features from those sequences (such as duration, distance, mean velocity, move efficiency, etc.) as Chu et al. (2018) did, we decided to take advantage of the exceptional performance of the CNNs in detecting patterns in images and process the mouse trajectories as images (i.e., 2-dimensional matrices). Thus, each sequence of mouse points is presented as a 2-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. This technique was also proposed in Wei et al. (2019), a research that was done in parallel

with us. Details of the classifier are presented in Section 3.2.2.

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 they 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.

Fusion of Detection Models

As discussed before, the framework performs a decision level fusion to leverage the complementarity between the two modules (i.e., the module that uses web logs and the module that uses mouse movements) based on the different levels of granularity that they provide which 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 argue that detection based on mouse movements can be less susceptible to evasion. Furthermore, in Section 3.1 we have shown that web log based detection is not very effective in the case of advanced web bots.

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 the two detection modules are combined. The equation to calculate the final score is presented below

$$score_{tot} = \begin{cases} score_{mv}, & \text{if } score_{mv} \geq thres_h \text{ or } score_{mv} \leq thres_l \\ w_{mv} * score_{mv} + w_{wl} * score_{wl}, & \text{otherwise} \end{cases} \quad (3.1)$$

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 which 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.

3.2.2 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, as well as the deep learning architecture employed by the detection module that uses mouse movements.

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. The features used were the ones presented in Section 3.1 and include the most promising ones that have been proposed in literature from 2012 to 2017, and are also applicable to our setting. More specifically, the full list is the one presented in Appendix C, without features 9, and 13-15. Feature 9 was not calculated, since the web server that we performed the experiments on did not contain any PDF files. Features 13-15 were not calculated, since the experiments were conducted in such a way that users access the server directly - thus all sessions (both human and bot ones) had the same values in those features.

The next step concerns the selection of the classifiers to be employed in our voting scheme. As discussed in Section 3.1 there is no clear performance benefit of one algorithm over the others, thus the most popular classification algorithms were selected, which are the SVM, the Random Forest, the Adaboost, and the MLP classifiers. Additionally, these classifiers construct an ensemble classifier (Voting classifier) that performs a class probability averaging of all the available classifiers. 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 (as discussed in Section 3.1). For the implementation of these algorithms the scikit-learn⁷ Python library was used.

⁷<https://scikit-learn.org/stable/>

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, which is relevant to our problem. Similar research that was done in parallel with us, such as Wei et al. (2019), has also used CNNs.

Inspired by the architecture used in Wei et al. (2019), as well as considering the nature and complexity of our problem (i.e., detecting line patterns), we chose the architecture presented in Table 3.8. For the implementation of the Deep Neural Network (DNN), the Keras⁸ Python library was used.

Table 3.8: 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

3.2.3 Evaluation

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, the dataset, the evaluation metrics that we considered, the configuration of the employed classification algorithms, and, finally, the configuration of the web bots that we used for the experiments.

Evaluation Methodology

The purpose of this work is to examine the effectiveness of the web bot detection framework when faced with malicious advanced web bots. For our experiments we consider a

⁸<https://keras.io/>

very common category of advanced web bots that is presented in Chapter 2 which aims 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.⁹

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 Iliou et al. (2021a). 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.

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 which 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 3.2.1, we

⁹<https://www.wikipedia.org/>

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 3.2.1); thus, no retraining is required. On the other hand, 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.

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 that 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.

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 consisting of the authors of Iliou et al. (2021a) was 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.

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 5 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 Iliou et al. (2021a); 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 9-20 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.

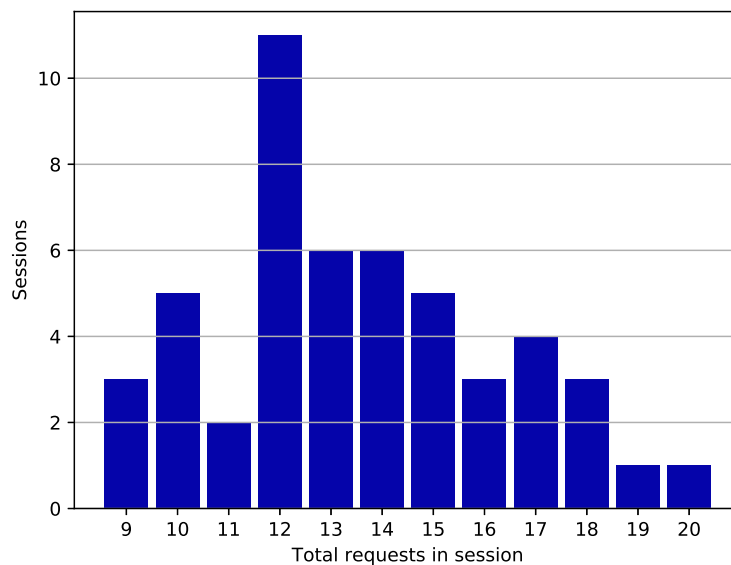


Figure 3.11: Distribution of total requests per session for the first phase of experiments

The distribution of the total number of requests of the human sessions for the first phase of experiments is presented in Figure 3.11. 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).

In the first phase of the experiments, the framework was evaluated on two datasets, (i) the D3 which contains the human sessions and the moderate web bot sessions, and (ii) the D4 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 period. The dataset details are presented in Table 3.9 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 3.9: Human, moderate and advanced web bots sessions and total requests for all sessions of the first phase of experiments (sessions/requests).

	Humans	Bots		D3	D4
		Moderate	Advanced	(Humans + Mod.)	(Humans + Adv.)
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

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. Additionally, we followed the appropriate procedures for the collection of the data, where a research ethics checklist was created to evaluate the risk of the research, and the human subjects were provided with a participant information sheet including several information about this research (e.g., what this research is about, how their data will be used, etc.), and a participant information sheet that each human subject had to sign before participating. These documents are included in Appendix D.

The distribution of the total number of requests in the sessions of the additional 28 users is presented in Figure 3.12. 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.

In the second phase of the experiments, the framework was tested on two datasets, (i) the D5 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 D6 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 D6 was that in a real-world scenario, such web bots

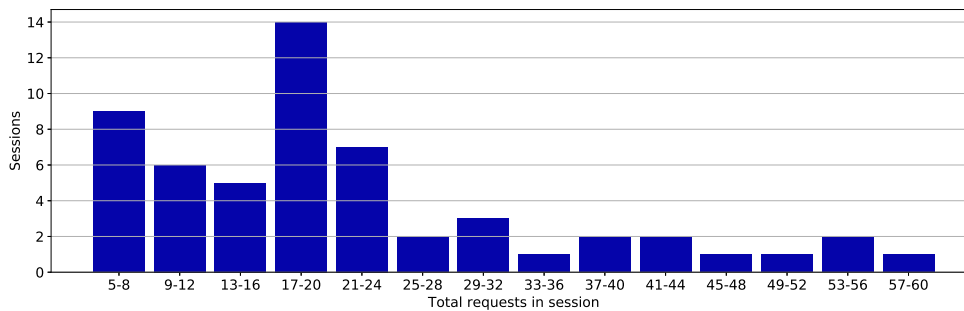


Figure 3.12: Distribution of the total requests per session for the second phase of experiments

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 D6 we considered 28 moderate and 28 advanced web bot 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 were 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 7-fold cross validation at user level. More specifically, we split the dataset into 7 folds, each fold containing the sessions of four users (thus, each fold containing 8 human sessions in total, as each human visitor made 2 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 library¹⁰), 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 3.10.

By comparing our datasets with those used in literature, we see that there are works with a number of users that is close to ours, such as Wei et al. (2019) where they used 50 human subjects, while other works have considerably more human subjects, such as Chu et al. (2018) with 1000 human subjects. However, getting that amount of users is not

¹⁰<https://keras.io/getting-started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development>

Table 3.10: Human, moderate and advanced web bots sessions and total requests for all sessions of the second phase of experiments.

	Humans	D5		D6	
		Adv.	Hum.+Adv.	Mod.+Adv.	Hum.+Mod.+Adv.
sessions	56	56	112	56	112
requests	1211	754	1965	734	1945

easy. These datasets were not public and we were not able to get access to them. Since the purpose of this chapter is to show the benefits of combining web logs with mouse trajectories for the detection of web bots as opposed to using individual approaches (with all those tested on the same dataset) and not to directly compare this approach with others in literature, we argue that the differences in the size of the datasets do not play a critical role in that. Had we wanted to compare the performance of our approach with other approaches in literature, a bigger dataset should have been used.

Evaluation Metrics

The evaluation metrics that were used are similar to the ones presented in Section 3.1.2. Specifically, following the works of Rude and Doran (2015), Stevanovic et al. (2012), Wang et al. (2013), and Rovetta et al. (2017) we calculated the accuracy, a common metric used in the web bot detection problem. Furthermore, to gain a better understanding of our framework’s performance and by also following the works of Alam et al. (2014), Wang et al. (2015a), Doran and Gokhale (2016), Sisodia et al. (2015), Stevanovic et al. (2012), and Rovetta et al. (2017), we calculated the precision, recall, and F-score evaluation metrics, which are also commonly used for the evaluation of web bot detection frameworks.

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.

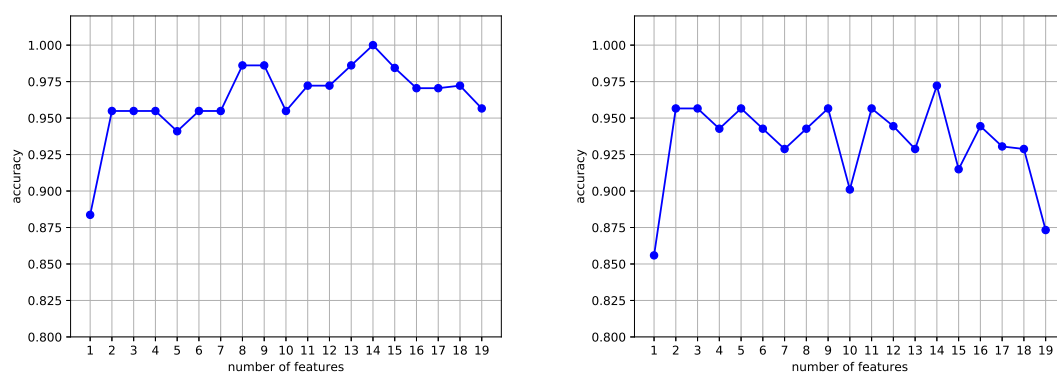
Classifiers Using Web Logs: This module employs an ensemble of four well established machine learning algorithms: SVM, RF, AdaBoost, and the MLP Classifier. These algorithms have been selected, since they are the most common ones in web bot detection literature, as discussed in Section 3.1.2. Naive Bayes is also very common, but we opted

from using it because of its very low performance in some works such as Stevanovic et al. (2012) and Sisodia et al. (2015).

For the selection of the parameters of these four classifiers, similarly to Section 3.1, we performed an exhaustive search over specified parameter values (grid search) and chose the ones with the highest accuracy using a 2-fold cross validation on the training data. 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. The final values of the parameters are presented in Table 3.11.

To select the best performing features for our voting (ensemble) classifier, similarly to Section 3.1, we used the SFS technique, a greedy algorithm that can be used to reduce the feature space presented in Pudil et al. (1994). More specifically, we used the Sequential Forward Floating Selection (SFFS) technique presented in Pudil et al. (1994), 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.

Figure 3.13: Sequential Forward Floating Selection for D3 (left) and D4 (right)



The accuracy of the SFFS for both the D3 and D4 datasets is presented in Figure 3.13. Based on this, we selected the set of features that resulted in the highest accuracy in the training set for both the D3 and D4. For both the D3 and D4 the number of features that present the highest accuracy are 14. These selected feature indexes are $\{0, 1, 3, 4, 6, 8, 11, 12, 20, 17, 18, 19, 21, 22\}$ for D3 and $\{0, 2, 3, 4, 5, 6, 7, 8, 11, 12,$

Table 3.11: The parameters of classification algorithms that use web logs.

Classification Algorithm	D3: Humans - Moderate bots	D4: 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=10 ⁻³ , b1=0.1, b2=0.1, e=10 ⁻⁸ , hidden layer sizes: (100, 50), constant learning rate	ReLU activation, adam solver, a=10 ⁻³ , b1=0.9, b2=0.9, e=10 ⁻⁸ , hidden layer sizes: (100, 50), constant learning rate
Random Forest	estimators=200, Gini crit., max features= $\sqrt{\#features}$, min samples per leaf = 1, min samples split = 2, max depth=10, out-of-bag samples used	estimators = 200, Gini crit., max features= $\sqrt{\#features}$, 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

18, 19, 20, 22} for D4 (see Appendix C for the corresponding features).

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 in order to further increase the robustness of our approach.

Classification Using Mouse Movements: As discussed before, this module utilises a DNN

architecture that utilises CNN layers for the detection of web bots based on their mouse trajectories. After a manual inspection of our datasets, and to account for the high memory usage of the input matrices (1080×1920), 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.

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 means that the visitor is very likely to be a human. The framework combines these scores as shown in Equation 3.1. 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 D4. 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 we consider the session to be a bot. Otherwise we consider the session to be human. These values were selected heuristically.







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. As discussed by Distil Networks (2018), malicious web bots can vary from simple scripts to bots that automate browsers and present a humanlike behaviour. 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 browser-like 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, such as the works of Iliou et al. (2017), Campobasso et al. (2019) and Pastrana et al. (2018), as well as in reports of web bot detection companies, such as Distil Networks (2019). 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 on the other hand exhibit some intelligence by performing a heuristic hyperlink selection, as well as some more advanced mouse movements based on the web pages' content. Table 3.12 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 below.

Table 3.12: 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			

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 9-20 (see Section 3.2.3). 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 which 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.

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 9-20 (see Section 3.2.3). 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 and following the work of Campobasso et al. (2019), 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.

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

$$lines_to_read = \frac{content_length - template_length}{length_to_lines} \quad (3.2)$$

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 which 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 3.12 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 3.2. 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.

Software for Generating the Web Bots: For the purposes of this work, we generated the web bots using the Selenium¹¹ 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 software in its default configuration can be detected in a deterministic way, using advanced fingerprinting techniques. Thus, in a real world scenario, Azad et al. (2020) have shown that further configuration on those tools can be applied to avoid detection based on their fingerprint. Additionally, Akrouit et al. (2019) proposed the use of real browsers instead of browsing automation software, resulting in fingerprints that are indistinguishable.

¹¹<https://www.seleniumhq.org/>

3.2.4 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, and the second phase, where the framework is evaluated in a more real-world scenario, where suspected moderate and advanced web bots cannot always be isolated before being passed to the detection models.

Regarding the first set of experiments, we initially present the overall performance of our framework using the D3 dataset when the modules are used alone or in combinations. 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. Finally, in the second set of experiments, we evaluated our framework over time on the sessions generated by the additional users.

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 D3 and D4 (see Section 3.2.3). The results of our framework are presented below.

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 3.1. The results are presented in Table 3.13.

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

Table 3.13: Evaluation of the web bot detection framework per session for D3 and D4

		web logs	mouse traject.	web OR mouse	average	fusion
		D3 / D4	D3 / D4	D3 / D4	D3 / D4	D3 / D4
Bot class	<i>Precision</i>	0.93 / 0.92	1.00 / 0.88	0.94 / 0.83	1.00 / 1.00	1.00 / 1.00
	<i>Recall</i>	0.87 / 0.80	1.00 / 1.00	1.00 / 1.00	1.00 / 0.93	1.00 / 1.00
	<i>F-score</i>	0.90 / 0.86	1.00 / 0.94	0.97 / 0.91	1.00 / 0.97	1.00 / 1.00
Human class	<i>Precision</i>	0.88 / 0.82	1.00 / 1.00	1.00 / 1.00	1.00 / 0.94	1.00 / 1.00
	<i>Recall</i>	0.93 / 0.93	1.00 / 0.87	0.93 / 0.80	1.00 / 1.00	1.00 / 1.00
	<i>F-score</i>	0.90 / 0.88	1.00 / 0.93	0.97 / 0.89	1.00 / 0.97	1.00 / 1.00
<i>Accuracy</i>		0.90 / 0.87	1.00 / 0.93	0.97 / 0.90	1.00 / 0.97	1.00 / 1.00

higher accuracy and F-score than the one that uses web logs. Concerning the precision and recall (for the bot class), we see that in D3 the detection module that uses mouse movements achieves 100% in both evaluation metrics while the detection module that uses web logs has lower precision and recall (which indicates the effectiveness of the mouse movements against moderate web bots). In D4 we see that the detection module that uses mouse movements achieves higher recall but lower precision when compared with the one using web logs. This indicates that mouse trajectories can be used to identify all web bots, but may misclassify some humans as bots, while web logs are better at not misclassifying humans as bots but fail to detect all bots.

Concerning 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 3.1, 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.

By comparing our results with literature, we see that the performance of our detection module that uses web logs is slightly worse than those presented in literature, while the performance of the detection module that uses mouse trajectories is similar to those

from literature. Specifically, the approaches that use web logs in literature and consider simple and moderate web bots (there are no works in literature that consider only moderate web bots) achieve an F-score and accuracy that goes up to 0.91 in Buehrer et al. (2009), and 0.958 in Almahmoud et al. (2019), while Grzinic et al. (2015) achieved an F-score that goes up to 0.997. These performances are higher compared to ours, which can be attributed to the fact that literature uses both simple web bots and moderate web bots (with simple web bots being more easily detected). Concerning the mouse trajectories, we see that works in literature consider only moderate web bots, and our performance is close to theirs. For example, Chu et al. (2018) achieved a recall higher than 0.996 and an AUC more than 0.999. These are comparable with our F-score and accuracy when detecting moderate web bots, which are 1.00 each. Thus, we see that our results are in line with the ones from literature.

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 the previous section, 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 3.14.

Regarding the performance of the detection module that uses mouse movements against moderate web bots (i.e., D3), as well as the average and fusion modes, we see that it achieves a very high accuracy, precision, recall, and F-score making only a few misclassifications in some of the first requests and making no misclassifications after eight requests, showcasing the effectiveness of mouse movements against moderate web bots (as also shown in Table B.2 from Appendix B, which presents the respective performances from other works in literature). Concerning the detection module that uses web logs as well as the OR combination of the two detection modules, we see that the accuracy and F-score range between 0.8 and 1.0, with the OR combination being affected by the performance of the web logs detection module (i.e., errors from the web log module are propagated to the OR combination). Also, we see that the OR combination achieves a very high recall of 100% (at the expense of the precision), which indicates that all the bots were retrieved, but also some humans were misclassified (something that is unwanted in a realistic scenario).

Also, as expected, we see generally a lower performance of the framework against

Figure 3.14: Performance over requests for D3 (left) and D4 (right)



advanced web bots (i.e., D4) compared to moderate ones (i.e., D3). In D4, we see generally a high performance in the first two and three requests (which is close to D3), which decreases for a few requests and then increases again. The increase in performance in the first requests and the following drop has also been observed in Cabri et al. (2018) as well. 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. Thus, detection approaches can only examine the first requests in a session to identify web bots, instead of waiting for the session to finish, or at least waiting a few requests before performing the model prediction. Even though the framework manages again to achieve a high performance after a few more requests (e.g., after the 13th request), allowing bots to perform a few requests before detecting them (i) is not very efficient, and (ii) can also be dangerous (depending on the bot type).

Concerning the individual performances of the different detection modules in D4, we see that regarding accuracy, F-score, and recall, the module that uses web logs achieves the lowest performance, affecting also the OR combination. The module that uses mouse movements performs better compared to the one that uses web logs, and the weight and fusion combinations achieve very high performance, with the latter achieving 100% to all evaluation metrics after the 13th request. While the modules that combine web logs and mouse movements (i.e., OR combination, average, and fusion) achieve a similar behaviour regarding the precision evaluation metric, we see that the web logs perform better. This indicates that the detection module that uses web logs is better in not misclassifying human sessions as bots, while the detection module that uses mouse movements is better at detecting all bots (in the expense of misclassifying some human sessions). Thus, even though the detection module that uses mouse movements performs generally better than the one that uses web logs, the latter is useful when we do not want to misclassify human sessions.

The aforementioned results outline the importance of combining the two detection modules, as opposed to using each one of them individually. Since the two modules model different temporal and spatial characteristics of the human behaviour, we see that they misclassify different sessions. Thus, their combination results in a better overall performance.

Also, there is no work in literature that considers moderate or advanced web bots and calculates the performance metrics over the requests. Thus, we cannot compare the results per request with other works. However, as discussed above, we see that the overall performance of our framework is in line with the literature.

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 $\alpha = 0.01$. For the D1, the difference is statistically significant in all cases except from (i) the “mouse movements”, “average”, and “fusion” in all evaluation metrics, and (ii) the “web logs” and “OR” combination in precision and recall. For the D2, the performance

difference is statistically significant in all cases except from (i) the “average” and “fusion” (with a p-value very close to being statistically significant) in F-score, (ii) the “web logs” and “mouse movements”, “web logs” and “OR” combination, and “average” and “fusion” in precision, and (iii) the “mouse movements” when compared with the “OR” combination, the “average”, and the “fusion” in recall.

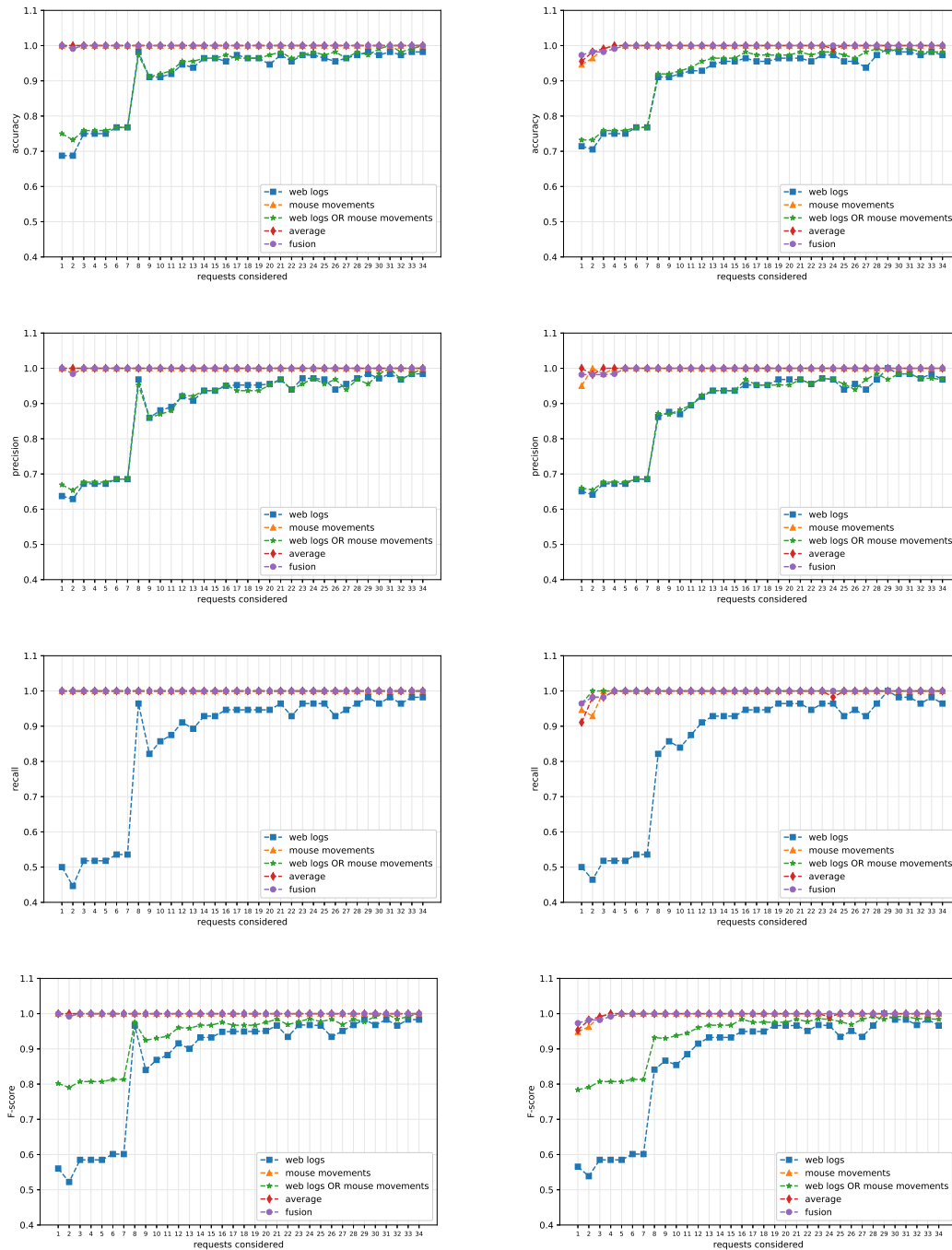
Second Phase of Experiments

In the second phase of the experiments, the average accuracy, precision, recall, and F-score were calculated per request across all iterations (see Section 3.2.3). In Figure 3.15 we present the results.

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. In D5, we see that the detection module that uses mouse movements as well as the “average” and “fusion” combinations achieve 100% to all evaluation metrics in the very first requests (but also do not drop, as they did in D4). The detection module that uses web logs also achieves a higher performance in D5 compared to D4 in all evaluation metrics except from the precision where in D4 it was slightly better. 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. Also, we see that the recall of the “OR” combination, in contrast to the other evaluation metrics of the “OR” combination, is very high, something that is expected since the OR module aims at classifying visitors as bots if either of the modules detects them as bots, resulting in detecting as many bots as possible.

The variety of the datasets is also evident in D6, where by mixing moderate with advanced web bots, there is an increased heterogeneity in the possible behaviours of web bots and this negatively affects the performance of the individual detection modules. When comparing D5 with D6, we see that in D6 the performance of all detection modules drops. For the detection module that uses mouse movements, as well as the “average” and “fusion” modes, we see a drop in the first requests in all evaluation metrics. After the first five requests, we see that the detection module that uses mouse movements,

Figure 3.15: Performance over requests for D5 (left) and D6 (right)



as well as the “average” and “fusion” modes manages to achieve 100% to all evaluation metrics (similarly to D5). For the detection module that uses web logs as well as the “OR” combination (in which errors from the web log module are propagated to) we see a similar but also lower performance between D5 and D6. Specifically, we see that in the first eight requests the performance is relatively low for all evaluation metrics, with the recall of OR combination not being affected (for the same reasons as in D5).

Furthermore, we see that the performance of the detection modules and their combinations stabilises after around 34 requests, which is why we opted to not show the performance of the framework when additional requests were used (i.e., after the 34th request).

Again, there is no work in literature that considers moderate or advanced web bots and calculates the performance metrics over the requests. Thus, we cannot compare the result of the second evaluation phase with other works. However, since the overall performance of the two evaluation phases is similar, as discussed in the first evaluation phase, our results are in line with the literature.

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 $\alpha = 0.01$. For the D5, the performance difference is statistically significant in all cases except from (i) the “mouse movements”, “average”, and “fusion” in all evaluation metrics, (ii) the “web logs” and “OR” combination in accuracy, precision, and F-score, and (iii) the “mouse movements” and “OR” combination in recall. For the D6, the performance differences are statistically significant in all cases except from (i) the “mouse movements”, “average”, and “fusion” in precision, recall, and f-score, (ii) the “web logs” and “OR combination” in accuracy and precision, and (iii) the “OR” combination and “average” in recall.

3.2.5 Discussion

In Section 3.1 we show that advanced web bots are utilised by malicious actors to evade detection. At the same time, we also show that even though the web bot detection techniques proposed in literature at that time were highly accurate for simple web bots, they perform poorly when tested on advanced web bots. To address these problems, we proposed a novel web bot detection framework in Section 3.2 which can accurately detect advanced web bots. It is an amalgamation of two detection modules, one which extracts information from web logs to determine if a visitor is a human or a web bot and one which 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. Furthermore, we show that the framework was able to detect web bots using only a few requests (i.e., not waiting for the session to finish). This is very important since it is common in literature to wait for the whole session (or at least allowing bots to make several requests before performing the prediction) which is not very effective and it may even be dangerous depending on the types of web bots considered.

Additionally, we examined the performance of the framework against web bots of different 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) to more effectively evade detection. However, the creation of such web bots should be challenging, since the more advanced a web bot is, the more complicated behaviour must be generated. Thus, this work raises the question of whether it is possible to create a set of evasive behaviours for advanced web bots effortlessly. This is something that is investigated in Chapter 4 of this thesis.

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). 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.

Also, when applying the proposed web bot detection framework to a web server with numerous visitors, its effect on 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.

To sum up, in this chapter we show the limitations of the state-of-the-art web bots detection techniques against advanced web bots that try to evade detection and we proposed a novel way to address this. The limitations of state-of-the-art detection approaches were not very evident because primarily of the datasets and annotation process used in literature. This could have resulted in companies using such detection approaches and considering that they are safe (because of the advertised effectiveness of those approaches), while being vulnerable, especially to highly sophisticated web bots that, as discussed in Chapter 2, are the most dangerous. Thus, it is very important to show the limitations of state-of-the-art approaches and propose additional techniques or improvement directions, since, especially in the area of cybersecurity, most organisations consider that they are safe as long as they have put some security measures in place (that are advertised to work very well), while usually those security measures fail. Showcasing the limitations of such approaches motivates the organisations to further research and employ protection mechanisms to increase their security instead of simply adapting an approach that is advertised to work.

4 Advanced Evasive Techniques for Web Bots

Advanced web bots can evade detection by using a browser fingerprint, and exhibiting a humanlike behaviour. For the latter, web bots can take advantage of the recent advances in machine learning to achieve such a behaviour. Since state-of-the-art behaviour-based web bot detection techniques (presented in Chapter 2 and Chapter 3) examine the web pages that bots visit as well as the mouse movements that they perform, in this chapter we investigate the performance of those detection methods when faced with advanced web bots that use the recent advances in machine learning to evade detection. We opted to do that since, as discussed in Chapter 2, there are several works that use heuristics or distributions to model a humanlike behaviour, but only a few that use machine learning, even though machine learning has several benefits over the other approaches and has also shown very promising results. Additionally, there are recent advances in machine learning that have not been tested on this topic yet.

Specifically, in this chapter we consider two approaches that web bots can use to evade detection based on their behaviour: (i) to browse the server and update their browsing behaviour based on whether they have been detected or not, and (ii) to train models on human browsing data and use them to generate (synthetic) humanlike data. We consider these to be the two main approaches that web bots can use when combined with machine learning, since (i) bots can either know nothing about the behaviour of the humans on a web server (and thus, bots have to find evasive behaviour through trial-and-error), or (ii) they can have access to human data from the target web server and use them to train machine learning models to generate humanlike behaviours.

The first approach has the benefit of not requiring any prior knowledge of the behaviour of humans on that server, since, as discussed in Chapter 2, it is not always easy to have this knowledge. However, this approach may require several unsuccessful attempts by the bots to find evasive behaviours. This approach has been proposed in a similar context by Akrouf et al. (2019), where web bots were trained using RL to generate humanlike mouse movements to bypass Google reCAPTCHA v3. In a similar manner,

we argue that web bots can use this approach to find evasive behaviours. The second approach (i.e., using human data to generate synthetic humanlike data) is commonly used in literature to generate evasive behaviours, but, as discussed in Chapter 2, usually is used for selecting the parameters of the distributions used to model a humanlike behaviour. Thus, instead of selecting the distributions' parameters, we can train models to generate humanlike behaviours, similarly to Acien et al. (2020b 2021). The generated synthetic data can be used by web bots to browse the web server in a humanlike manner. However, to achieve that, we need a database of human data to train our models.

For the first case (i.e., web bots updating their behaviour by interacting with the web server), web bots can use Reinforcement Learning (RL), which has shown exceptional performance in different domains with similar characteristics, including defeating professional players in games, such as the board game Go as shown in Silver et al. (2016). In RL, an agent (here, a web bot) performs some actions in an environment (e.g., visiting specific web pages) in order to maximize the notion of cumulative reward (in our case, to evade detection while browsing the web server). The web bots can update their behaviour based on, among others, whether they are detected or not.

For the second case, one of the recent machine learning techniques that has shown very promising results in generating new data with the same statistics as the ones used for training are the Generative Adversarial Networks (GANs), proposed in Goodfellow et al. (2014). GANs are architectures where two neural networks, the Generator and the Discriminator, contest with each other and are trained simultaneously in an "adversarial" setting. The Generator constantly tries to generate data that can fool the Discriminator into thinking that those data are real (and not synthetically generated). During training, the Generator progressively becomes better at creating data that look real, while the Discriminator becomes better at finding them. Specifically, as shown in Radford et al. (2016), GANs can be used for generation of very realistic images, which makes them ideal for the generation of humanlike mouse and touchscreen trajectories. Web bots can train GANs using mouse movements from human visitors, and then use these trained GANs to generate synthetic humanlike data that share similar characteristics with the ones used for training. Thus, in that setting, GANs are a very promising approach. Even though GANs have been used in Acien et al. (2020b 2021) to generate evasive web bot behaviours, they have only been used to generate mobile humanlike touchscreen trajectories and also they process the data as sequences instead of images. Thus, our approach is a novel one and utilises the capabilities of GANs to generate realistic images.

The generated data can then be used by web bots to browse the web servers and evade detection due to their humanlike behaviour.

In this chapter, we evaluate these two novel approaches for creating advanced web bots against state-of-the-art web bot detection techniques. More specifically, in Section 4.1 we present and evaluate the web bots that use RL to evade detection based on their browsing behaviour (i.e., the web logs that they generate), and in Section 4.2 we present and evaluate the web bots that use GANs to evade detection based on their mouse movements. RL and GANs are not the only options for the generation of evasive behaviours, but are good candidates for our purposes based on the unique characteristics of those methods and the nature of our problem (i.e., the generation of a humanlike browsing behaviour, and the generation of humanlike mouse movements). The main aim of this chapter is not to present the most effective and/or efficient approaches for generating evasive humanlike behaviours, but to show the possibility of using recent advances in machine learning to do so, by evaluating two recent machine learning based techniques that are well fitting for our settings.

4.1 Evasive Web Bots Using Deep Reinforcement Learning

Evasive web bots can be trained to evade detection, by trying different browsing behaviours and following the most promising ones. Specifically, they can start from a specific behaviour and try different approaches in regards to the web pages that they visit, the time that they spend on each web page, the sequence of web pages that they should follow, etc. in an attempt to avoid detection. After finding a behaviour that can be used to remain undetected, web bots can use it until the web server updates its detection models to include these new malicious behaviours to its detection models. This process can be repeated several times, where both parties (i.e., the detection module, and the web bots) update their methods. This approach has the benefit of not requiring any prior knowledge of the human behaviours on the target server, something that, as discussed in Chapter 2, can be problematic. However, several unsuccessful attempts might be required by the bots to find such behaviours.

One of the most promising machine learning methods that can be used for this type of problem (i.e., learning through trial-and-error) is Reinforcement Learning (RL). As discussed in Kaelbling et al. (1996), in RL, an agent performs some actions in an environment in order to maximize the notion of cumulative reward. Through interacting

with the environment, the agent can find a policy (i.e., the way that it makes decisions on what actions to perform) that results in a high reward.

In this thesis, we show that RL can be used by web bots for finding a policy (i.e., what web pages to follow and how much time to spend on each one of them) that allows them to remain undetected. The web bots continuously interact with the web server and update their behaviour based on, among others, whether they have been detected or not.

Even though RL has shown very promising results in different scenarios with similar characteristics as the web bot detection/evasion problem, including defeating professional players in games, such as the board game Go Silver et al. (2016), to the best of our knowledge only one relevant work exists in literature. Specifically, Akrouf et al. (2019) proposed the use of RL to bypass Google reCAPTCHA v3 by training the web bot to learn how to move the mouse and click on the reCAPTCHA button. Thus, to the best of our knowledge, this is the first work that examines the case of web bots using RL to evade detection from the web logs that they generate.

Next, we present a novel way that web bots can use RL to evade detection based on web logs and examine the effectiveness of state-of-the-art web bot detection techniques against those bots. For that, we initially present the mapping of the main RL concepts to the web bot detection/evasion problem (Section 4.1.1). Then, we present the details of the RL environment and the novel evasive web bots that use RL (Section 4.1.2). After that, we present the evaluation methodology and the experimental setup (Section 4.1.3), and the results of the evaluation (Section 4.1.4).

4.1.1 RL Main Concepts for the Web Bot Detection/Evasion Problem

As discussed before, in RL the agent performs actions in an environment in order to maximise the notion of cumulative reward. There are five main “concepts” in RL that should be mapped to the web bot detection/evasion problem: *agent*, *environment*, *action*, *state*, and *reward*. Following an intuitive approach, we map those concepts to the web bot detection/evasion problem, as presented in Figure 4.1.

Specifically, the web bots repeatedly browse the web server following different behaviours, while also examining whether they appear to have been detected. Based on whether they have been detected or not, web bots update their browsing behaviour. Since web bots are detected based on their browsing behaviour (i.e., the web pages that they visit), we consider that they have full observability of the environment (i.e., they know the exact states that they are in, which is calculated based on the web pages they have vis-

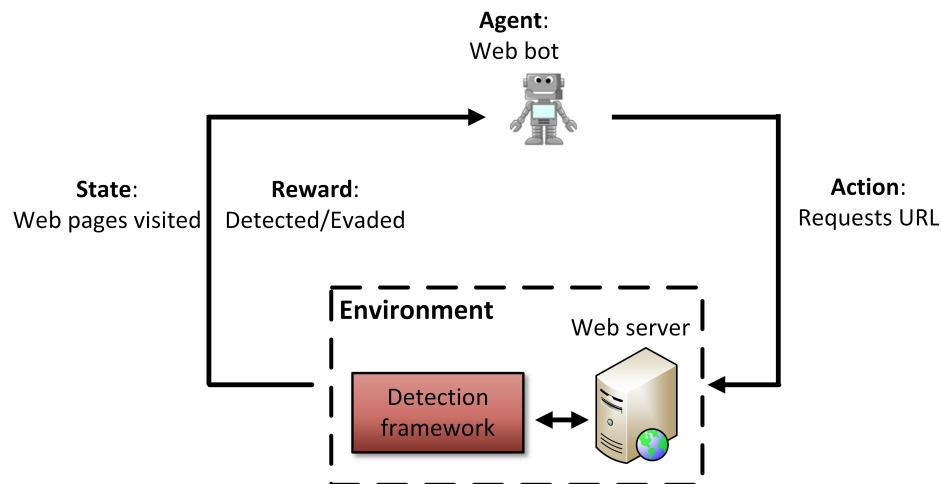


Figure 4.1: RL concepts

ited). In Table 4.1 we summarise these main RL concepts and how we instantiated them in the web bot detection/evasion problem.

Also, we assume that web bots can change their fingerprint when being detected, thus starting in a clean-slate every time (as far as the web server is concerned). This is not an unreasonable assumption, since, as discussed in Chapter 2 advanced web bots may have several unique fingerprints in their possession allowing them to change their fingerprint and be considered as new visitors by the web server if needed.

4.1.2 Evasive Web Bots Using RL

Web bots can take advantage of RL (which has shown very promising results in similar domains) to find behaviours that can evade detection. For that, we initially have to define (i) the appropriate *environment* (i.e., the web server with the bot detection framework), and then model our evasive web bots by defining (ii) the possible *actions* that web bots can perform, (iii) the *states* the web bots can be in, (iv) the *rewards* that the web bots will receive, and (v) the *model* (i.e., “brain”) of the *agent* (i.e., the web bot), responsible for deciding which action to perform at each state.

Environment

The environment consists of the web server along with the web bot detection framework. The web server hosts several web pages, simulating a real one. The detection framework. The detection framework follows state-of-the-art web bot detection approaches that examine the visitors’ web logs, and is the one presented in Section 3.1. This frame-

Table 4.1: RL concepts and instantiation into the web bot detection/evasion problem

RL concept	Description	Web bot detection/evasion context
<i>Agent</i>	The “thing” that senses the environment and performs some actions	Web bot
<i>Environment</i>	The real or simulated “world” that the agent can interact with	Web server and detection framework
<i>Action</i>	What an agent can do in its environment	Visiting of a website
<i>State</i>	Different configurations of the environment that the agent can sense	The web pages that the agent has already visited
<i>Reward</i>	A numerical value received by the agent from the environment as a direct response to the agent’s actions	Value that depends on the web page visited and whether the web bot is detected or not

work accesses the web logs of a web server, extracts the different sessions of users and then calculates the values of several features to first train its models and then classifies new visitors as humans or bots.

The general architecture of the web bot detection framework is presented in Figure 4.2, and is based on the one presented in Section 3.1. The detection framework uses as input web logs and splits the data into sessions. As opposed to the detection framework proposed in Section 3.1.1, in this case the PHP session ID is available and can be used to split the web log data into sessions. Then, for each session, the values of the features that are utilised by the framework are calculated, with the full list being presented in Appendix C. From that list, we did not use the features 13-15, since the experiments were conducted in such a way that users access the server directly - thus all sessions (both human and bot ones) had the same values in those features. Finally, the generated feature vectors are used as input to the (trained) classification module and the classification score is retrieved, based on which the visitor is blocked or not. As presented in Section 3.1.2, the framework ensembles several classifiers (i.e., it performs a class probability averaging all the available classifiers); the details of the classifiers are presented in Section 4.1.3.

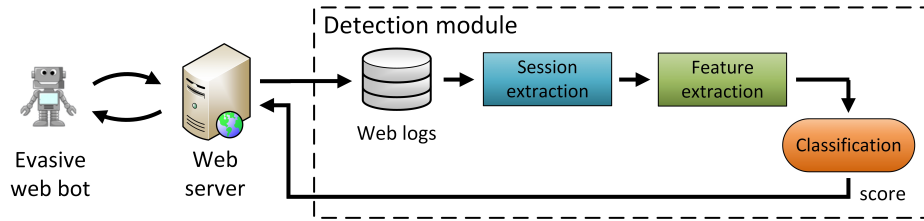


Figure 4.2: Web bot detection framework

Actions

The actions that the bots will be able to do should depend on what features the state-of-the-art web bot detection considers. As discussed in Section 3.1.1, these features examine the method and response of the HTTP request, the type of file(s) requested, and the browsing behaviour (including the pages visited and the respective time).

Thus, the different actions that are considered in this work have to do with (i) what web pages the bot can visit/download, and (ii) how much time it spends on them. Based on that and since web bots try to generate a behaviour similar to a human one, the proposed advanced web bots support the following general types of actions:

- *Simple download*: The web bot downloads only the main web page content (e.g., the HTML code) and not the additional files included (e.g., images, CSS, JavaScript, etc.).
- *Full download*: The web bot downloads the web page content and any additional files included (e.g., images, CSS, JavaScript, etc.).

Additionally, web bots can wait some time at each web page to simulate a “reading” functionality. We consider this waiting time to be between two values, the $time_{min}$ and $time_{max}$. Also, we consider those to be integers indicating the seconds that web bots will wait, since the respective features used by the web bot detection framework round the time in seconds, as shown in Appendix C.

Thus, the total number of different actions at each state is:

$$total_number_of_actions = N \cdot 2 \cdot (dt + 1) \quad (4.1)$$

where N is the number of web pages of the web server, the ‘2’ indicates the two modes of downloading (i.e., *simple download*, and *full download*), and $dt = time_{max} - time_{min}$ corresponds to the possible times that the web bot can “wait” on that web page (this is why we add one to the difference dt when calculating the total number of actions).

From Equation 4.1 we see that web bots can visit any web page when being on a specific web page (and not only the ones included in this web page), reflecting a more realistic scenario that does not limit the visitors to access only specific web pages. However, this assumes that web bots know all the URLs of the web pages of the web server. Assuming that web bots will repeatedly visit different web pages of the web server and based on the fact that the URLs of some web servers can be predicted or extracted using different methods (such as using search engines to gather relevant URLs), this is not an unreasonable assumption.

States

States are considered as the different configurations of the environment that the web bot can sense, and, in our case, are calculated based on the web pages that the bots download. Each action can result in changing the current state of the agent (i.e., web bot).

There are two ways to calculate the state in our case: (i) considering only the first time a web bot downloads a specific web page, or (ii) considering every time a web bot downloads a specific web page. The first approach would limit the intelligence of the agent by omitting information that has to do with going back and forth to the same web pages (something that humans usually do). The second case could result in an infinite number of possible states, since a web bot can visit a web page infinite times.

Thus, in our case we follow the second approach (i.e., considering every time a web bot downloads a specific web page) by adding an upper limit in how many times we consider downloads of the same web page. After that, we ignore any additional visits to this page, which could have resulted in an infinite number of possible states.

Additionally, a web bot can simply download a web page (which is common for simple scripts), or download also the additional sources included in this page (such as JavaScript files, CSS files, images, etc.). In the case of human visitors (i.e., where a browser is used), the latter (i.e., downloading additional sources included in the web page) is usually done only the first time each file is encountered and the downloaded files are cached on the browser.

Thus, in our setup we consider two separate states for each web page based on the way the web page can be visited: (i) only the web page is downloaded, and (ii) the web page is downloaded along with the additional files included in that page.

We define the state vector for a web server with N web pages as:

$$state_vec = [\#page_0, \dots, \#page_{N-1}, \#page_N, \dots, \#page_{2N-1}] \quad (4.2)$$

where $\#page_i$ is an integer number indicating how many times the agent has downloaded a specific web page $page_i$, with values between 0 and M , with M being the upper limit for the number of times that we consider that the web bot can download the specific web page. Additionally, web pages whose index differs by N (i.e., $page_i$ and $page_{N+i}$, $0 \leq i \leq N - 1$) correspond to the same web page, with the one with the low index indicating that only the web page is downloaded while the one with the high index indicating that, besides the web page, the additional files included in the web page (e.g., JavaScript, CSS, etc.) are also downloaded.

Rewards

Web bots are trained to maximise a notion of cumulative reward, received by the environment as a direct response to their actions. The selection of rewards is crucial since they guide the bot in achieving the wanted behaviour. For example, simply giving positive rewards to the web bot when it is not detected might result in the web bot staying in the same web page, as long as it remains undetected.

In our case, the target goal of the web bots is twofold: (i) to generate a behaviour that evades detection, and (ii) to explore the web server and visit new web pages. Thus, we consider the following types of rewards:

- *New web page reward*: Web bot downloads a new web page and does not get detected;
- *New state reward*: Web bot changes state (but does not download a web page that it has not downloaded before) and does not get detected (as previously discussed, a web bot can download the same web page up to M times resulting in another state each time);
- *Detection reward*: Web bot tries to download any web page and gets detected; or
- *Detection evasion reward*: Web bot manages to download M web pages and does not get detected.

Based on the above, the *detection evasion reward* should be the highest one, motivating web bots to follow behaviours that will allow it to evade detection. The *new web*

page reward should come next, motivating web bots to explore the web server, instead of staying at the same page as long as they remain undetected. The *new state reward* should follow, since web bots might have to re-visit a specific web page to evade detection since this is also something that humans do. Finally, the *detection reward* should be the lowest, and can also be negative, resulting in a “penalising” characteristic.

Agent

After having defined the environment, actions, states, and rewards, we have to choose the algorithm that the agents (i.e., web bots) should follow to learn what action to perform at each state.

We decided to base our *model* on the well-known Q-learning algorithm. Q-learning generates a matrix containing all possible states and the respective actions and a value for each state-action pair representing how useful a specific action is on a specific state based on the future reward that the agent will receive. This is called the action-value function, $Q(s, a)$. Using Q-learning, the web bot can find a *policy* (i.e., the way it makes decisions for what actions to perform at each state) to follow that maximises its future reward.

The $Q(s, a)$ is initialised randomly. Then, for each step t the agent takes an action a_t on a state s_t resulting in changing its state to s_{t+1} and receiving a reward r_t . Based on this transition, the $Q(s, a)$ is updated using the following equation:

$$Q^{\text{new}}(s_t, a_t) = \underbrace{Q(s_t, a_t)}_{\text{old value}} + \underbrace{a}_{\text{learning rate}} \cdot \underbrace{\left(\underbrace{r_t}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \cdot \underbrace{\max Q(s_{t+1}, a)}_{\text{estimate of optimal future value}} - \underbrace{Q(s_t, a_t)}_{\text{old value}} \right)}_{\text{Temporal Difference}} \quad (4.3)$$

where the a (i.e., learning rate) determines to what extent the newly acquired information overrides old information, and the γ (i.e., discount factor) determines the importance of future rewards. To this end, the Temporal Difference (TD) learning approach is followed, where the agent tries an action (a_t) in a particular state (s_t), and evaluates its consequences in terms of the immediate reward and its estimate of the future rewards it will receive by moving to the next state, s_{t+1} .

As discussed above, to calculate $Q(s, a)$, the agent (i.e., web bot) interacts with the environment and updates $Q(s, a)$ based on the states it has been in and the actions it performs. However, by following only the most promising actions in a state, the agent chooses only the actions that give the highest immediate reward, which might be far less

than the future reward that it could have received by performing different actions (which are currently unknown).

Thus, the agent should find a balance between “exploitation” (i.e., performing the most promising action), and “exploration” (i.e. taking a random action, hoping for a better future reward). This is referred to as the “explore-exploit” dilemma, where the agent has to decide when to explore and when to exploit (i.e., take the best action based on the current knowledge of the environment). In principle, the agent could have always performed random actions to find the best behaviour to follow, but this would take a lot of time due to the agent following paths that are not very promising.

To calculate the $Q(s, a)$, our agent uses the Epsilon-Greedy approach, where it performs a random action or the most “promising” action on a current state based on a probability ϵ . Based on this probability, the agent either (i) follows the most promising states (in regards of the future rewards that it will receive), or (ii) performs random actions that may result in states with better future rewards.

To account for the fact that by following the Q-learning algorithm, the agent might take a lot of time to train (since there is a large number of actions the agent can perform, as shown in Equation 4.1), we used the Deep Q-Network (DQN) algorithm proposed by DeepMind Mnih et al. (2013). DQN uses a Deep Neural Network (DNN) to calculate the equivalent of the $Q(s, a)$ used in Q-learning that allows us to train the web bots faster. Thus, instead of calculating the $Q(s, a)$, we calculate an approximation using DNNs.

Specifically, in DQN, we approximate the $Q(s, a)$ by considering a network that takes as input the current state vector and have multiple output nodes, each one representing the value for each different action (i.e., the output layer of the neural network has the same size as the number of possible actions). Additionally, we use the experience replay technique, where we store all of the agent’s experiences and then randomly sample from these experiences (i.e., get a random minibatch of the transitions) and use the samples to perform a gradient descent to train the DQN.

As it is evident, the DQN architecture depends on the number of states (which depends on the web pages considered, as shown in Equation 4.2), and the total number of actions (which depends on the web pages and the waiting time, as shown in Equation 4.1). Since these depend on the experimental setup, the DQN architecture is presented in the next section.

4.1.3 Evaluation

To assess the effectiveness of the web bots that use RL to evade detection, we consider the case of scraping web bots that continuously visit a web server to harvest its content. The web server uses a web bot detection framework that examines the visitors web logs to identify and block bot visitors.

Evaluation Methodology

To simulate a more realistic scenario (where both the evasive web bots and the detection framework continuously update their techniques to achieve their goals), the evaluation takes place in two phases. In the first phase, we assume that the detection framework knows nothing about the web bots that use RL, while in the second the detection framework is trained on the behaviours of those bots.

In the first phase of the experiments, the same dataset used in Section 3.2 is reused here. As discussed in Section 3.2.3, this dataset includes both human and bot sessions, with the latter being generated from bots that use heuristics to evade detection. We opted to do that, to see how the web bots using RL can be trained to evade a bot detection framework that was trained on different types of bots.

In the second phase, we use data collected during the first phase from the web bots that used RL and successfully evaded detection to re-train the detection framework and evaluate how well these web bots can still evade detection. Additionally, we examine the case of web bots re-training their models to the re-trained detection framework. We argue that this setup simulates a scenario closer to a real-world setting, where both the defender (i.e., the web bot detection framework) and the attacker (i.e., the evasive web bots) continuously update their models.

Moreover, since in a realistic scenario we want to detect web bots as soon as possible, the detection framework examines the behaviour of the web bots per-request, and uses different classification models for each number of requests considered. Thus, to train the detection framework, we 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 reaches the maximum number of requests available, we stop increasing the number of requests considered for that session.

In both phases of the experiments, we performed the same train-test split (with about 70% of the data considered as training and the rest as testing) on the visitors when

evaluating the detection framework (i.e., we split the data based on the visitors to group multiple sessions from the same visitor in the same set). Additionally, at each request considered, we performed a grid search over several possible values of hyperparameters on the (current) training set and select the best performing ones using a 2-fold cross validation. Thus, each classifier uses a unique set of parameters for each number of requests considered. This increases the training time considerably but usually increases the performance.

Since in RL the agents (in our case the web bots) can be trained indefinitely, we evaluated them at different training times (i.e., at different numbers of requests performed cumulatively by all web bots). We opted to do that because when a bot is detected, it is blocked and will not be able to re-visit the web server (at least temporarily). Thus, there is a trade-off between how much training the web bots should do (in regards to the bots required and the time) and the evasiveness of the behaviour generated.

Finally, besides the web bots that use RL to evade detection, we also tested other types of web bots that use heuristics. We did that for comparison purposes and because of different approaches proposed in literature on evasive web bots that also use heuristics to evade detection, such as the works of Campobasso et al. (2019) and Iliou et al. (2017). The details of the configurations of the web bots considered are presented later in this section.

Dataset

As discussed before, for our evaluation, the same dataset used in Section 3.2.3 is re-used. In short, a web server that contains a total of 110 web pages from 11 topics of Wikipedia was used. Over the course of the experiments, 28 users were asked to visit the web server and create two sessions each, with each session being between 15-20 minutes. Thus, the total sessions generated were 56.

Additionally, we used the 56 sessions generated from the *advanced web bots* that were created for evaluation purposes in Section 3.2 for training. In short, those advanced web bots try to avoid detection by (i) selecting the hyperlinks to follow in a heuristic way in an attempt to make web bots stay at the same topic (i.e., follow web pages that belong to the same category), and (ii) simulating a reading function, and so they spent an amount of time at each web page based on the web page's content.

Bot and Environment Configurations

Web Bot Detection: The web bot detection framework from Section 3.1 follows state-of-the-art approaches and ensembles the most popular classification algorithms in this domain: SVM, RF, AdaBoost, and MLP. Also, it performs a class probability averaging of all the above classifiers to increase its effectiveness.

As discussed before, each classifier is trained on each number of requests considered, with the total number of ensemble classifiers being equal to the total number of requests considered. For each classifier, we re-calculate the set of hyperparameters that achieve the highest accuracy using grid search with a 2-fold cross validation. This was done to increase the general effectiveness.

Furthermore, we consider that the detection framework changes its response when a visitor is detected as a bot in a way that is recognisable by the bot (such as blocking any additional requests from that bot). However, this is not always the case, since some web administrations might simply prefer to monitor all the actions of the bots instead of blocking them or deliver custom content.

Bots Using Deep RL: As discussed in Section 4.1.2, to model the evasive web bots that use RL, we need to define the possible *actions* that web bots can perform, the *states* the web bots can be in, the respective *rewards*, and the *model* that the *agent* (i.e., web bot) will use to decide the actions to perform. The respective configurations are presented below:

Actions: For the actions, we heuristically selected the $time_{min} = 2 \text{ sec}$ and the $time_{max} = 10 \text{ sec}$ so that web bots can perform relatively rapid actions (to scrape the server within a short period of time), but not so rapid so as to risk detection.

States: As mentioned in Section 4.1.2, a maximum value of the times, M , that we consider that a web page has been visited is calculated. Thus, we set $M = 60$, which is the maximum number of web pages downloaded by the humans in our dataset.

Rewards: The rewards should motivate the web bot to (i) remain undetected, and (ii) visit new states. Based on our preliminary experiments, we saw that, as long as the reward values are in a specific order (with *detection evasion reward* being the highest one followed by the *new web page reward*, etc.), the RL bots can be trained to evade detection. The rewards in Table 4.2 were selected heuristically to focus on evasion, which we found to be more effective.

Agent: As discussed in Section 4.1.2, we use the DQL algorithm, with the input

Table 4.2: Rewards

Reward	Value	Reward	Value
<i>New web page</i>	+1.0	<i>Detection</i>	0.0
<i>New state</i>	+0.1	<i>Detection evasion</i>	+100.0

layer size being equal to the the number of states (i.e., $2 * N = 2 * 110 = 220$), and the output layer size being equal to the number of possible actions (i.e., $2 * N * (dt + 1) = 2 * 110 * (8 + 1) = 1980$). For the rest of the architecture (i.e., the hidden layers), seven different (one- or two-) hidden layer architectures have been selected heuristically to see how the different architectures affect the performance: N1 (128), N2 (256), N3 (512), N4 (1024), N5 (128, 256), N6 (256, 512), N7 (512, 1024). Concerning the activation functions of the layers, for the output layer, the linear activation function was used, and for all hidden layers the ReLU.

For the implementation of the environment, the Gym¹ Python library was used, that allows comparing different RL algorithms by providing a standard API to communicate between learning algorithms and environments, as well as a standard set of environments compliant with that API. For the implementation of the DQN, the keras-rl2² python library was used, which implements some state-of-the art deep RL algorithms and seamlessly integrates with the Keras deep learning library. Furthermore, keras-rl2 works with OpenAI Gym out of the box.

The hyperparameters for the DQN were selected heuristically and are presented in Table 4.3. The value of ϵ was selected to focus on following behaviours that were found to be evasive, but also, with a smaller probability, to try finding new promising behaviours (i.e., exploring). The γ was selected to focus on future rewards. For the other hyperparameters, commonly used values were selected.

Finally, since we want to evaluate how well the web bots perform when placed into different web pages as a starting point in testing (and not choosing the most evasive starting point each time), each bot performs a random step at the beginning³.

Bots Using Heuristics In addition to the web bots that use RL to evade detection, we

¹<https://github.com/openai/gym>

²<https://github.com/taylormcnally/keras-rl2>

³This was not supported by the keras-rl2 library, so we had to update the source code of the library

Table 4.3: Parameters and configurations

Parameter	Value
Warm up steps	100
Learning rate	10^{-3}
Epsilon greedy probability (ϵ)	0.2
Discount factor (γ)	0.9
Replay Memory	size=50k, window=1

used some additional types of web bots that use heuristics to evade detection, mainly for comparison purposes. These bots are detailed below:

- *Simple download bot*: Web bot that performs only *simple download* actions, i.e. downloads only the web page content and not the additional files included in the web page (e.g., CSS files, JavaScript files, images, etc.);
- *Full download bot*: Web bot that performs only *full download* actions, i.e. downloads the web page content and additional files included (e.g., CSS files, JavaScript files, images, etc.);
- *Random bot*: Web bot that downloads either only the web page or also the additional files included (i.e., it performs a random action from *simple download* or *full download*);
- *Heuristic bot #1*: Web bot that performs a *full download* for the first request and a *simple download* for the rest of the requests (thus simulating the behaviour of a browser, where the additional content of web pages is cached); and
- *Heuristic bot #2*: Web bot similar to *heuristic bot #1*, but the web bot waits for a time between 8~10 seconds (instead of 2~10, as discussed before in this section), since humans usually spend more time on web pages.

Evaluation Metrics

To evaluate the web bot detection framework, the evaluation metrics used are similar to the ones presented in Section 3.2.3. Specifically, we calculated the balanced accuracy, and the precision and recall for both classes (i.e., web bots and humans). To evaluate

how well the web bots evaded detection, we calculated the evasion percentage (i.e., how many web bots evaded detection from the total web bots tested).

4.1.4 Results

As discussed previously, the evaluation takes place in two phases; in the first phase the web bots that use RL are trained and evaluated on an already trained bot detection framework, and in the second phase they are evaluated on a re-trained detection framework based on the new bot behaviours collected in the first phase. In the second phase, (i) we evaluate the bots trained in the first phase, and (ii) we examine the case of web bots also re-training on the re-trained detection framework. Next, we present the results of the two phases.

First Evaluation Phase

In the first evaluation phase, we initially re-evaluate the web bot detection framework using the dataset from Section 3.2 (for completeness purposes), and then evaluate the evasive bots against this framework.

Web Bot Detection Performance: The performance of the detection framework per request considered is shown in Figure 4.3. In general, the framework can effectively detect web bots, achieving 100% to all evaluation metrics at nine requests. However, since web bots use heuristics to evade detection, for some specific requests (such as 20, 23, etc.) some bots achieved a behaviour very close to humans, resulting in misclassifying them. For most requests, we see that the precision for the human class was not affected, meaning that there were no humans detected as bots.

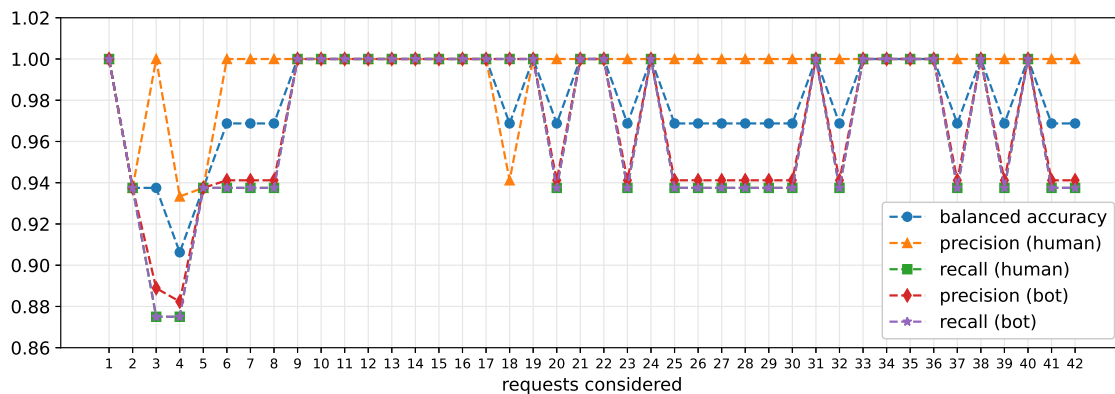


Figure 4.3: Performance of the web bot detection framework

Evasive Web Bots: The evaluation of the bots that use heuristics and RL are presented in Table 4.4 and Table 4.5 respectively. At each case, 1000 bots were used. For the RL bots, we tested different deep learning architectures (see Section 4.1.3), and we consider them being initially trained on the pre-trained web bots detection framework, and then using 1000 additional bots for testing.

Table 4.4: Performance of evasive web bots using heuristics

Bot type	Evasion percentage
Simple download	0.7%
Full download	0.0%
Random	0.0%
Heuristic #1	0.1%
Heuristic #2	0.0%

Table 4.5: Evasion percentage (%) of RL bots with different deep learning architectures

Requests (training)	N1	N2	N3	N4	N5	N6	N7
5k	5.0	1.3	1.1	0.0	0.0	0.0	0.0
10k	0.5	0.0	3.6	5.5	5.0	2.3	1.1
15k	0.0	1.8	0.9	10.9	1.4	6.8	3.4
20k	0.8	0.2	1.2	6.3	7.2	2.4	5.1
30k	7.7	0.9	7.5	38.5	5.8	18.3	11.0
40k	4.0	9.1	15.2	37.3	0.5	9.2	6.5
50k	10.6	10.9	27.6	36.2	15.2	3.7	9.2
100k	28.5	41.4	25.1	39.3	33.0	24.4	15.4
200k	32.4	20.7	30.7	22.6	30.7	34.5	37.1
300k	38.0	40.2	48.4	44.2	29.2	33.1	26.4
400k	46.0	48.2	44.2	31.9	46.9	37.4	32.4
500k	34.7	33.6	35.2	38.0	26.0	40.5	25.6

We see (Table 4.4) that the heuristic approaches tested can be easily detected. Only a few web bots, following mainly the *simple download* approach, evaded detection. This

can be attributed to the fact that the *advanced web bots* used for training the detection framework performed a *full download* in their first request so as to exhibit a humanlike behaviour.

On the contrary, web bots using RL were able to find behaviours that can evade detection (Table 4.5). Generally, the more requests considered in training, the better the evasive web bots perform, with some fluctuations. This can be attributed to (i) web bots trying (“exploring”) new actions to specific states (instead of following the most promising ones) which can result in the decrease of the reward values of those states (resulting in web bots no longer selecting them), and (ii) the detection framework being very “sensitive” to the behaviour of the visitor, meaning that small changes in their behaviour can reveal their bot nature. We also see that architectures with a single hidden layer work better (with N3 being the more evasive one), and the smallest of the two-layer architectures (i.e., N5) also achieves a high evasion percentage. These can be attributed to the fact that the evasive behaviour is not very complex, thus simpler neural networks with adequate neurons work better.

We stopped the training at 500k requests since (i) the web bots of most architectures had already achieved a very high evasion percentage (almost half the bots evade detection), and (ii) there is a trade-off between how many bots we use for training and what evasion percentage is achieved. For the latter, we needed more than 24k and 40k bots per architecture to train for 300k and 500k requests (respectively), with most of them requiring different fingerprints in a realistic scenario where each bot getting detected should change its fingerprint if it wants to revisit the web server shortly.

Second Evaluation Phase

In the second phase, we used the most evasive RL bots (i.e., the ones with the N3 architecture trained at 300k requests) to re-train the detection framework. Next, we first present the performance of the re-trained web bot detection framework and then we re-evaluate the web bots on the updated framework.

Web Bot Detection Performance: As shown in Figure 4.4, the re-trained web bot detection framework achieves a better performance compared to the first phase making very few mistakes on specific requests. This indicates that the web bots that use RL generated a behaviour that, even though it evaded detection, can mostly be distinguished from the human behaviour, when known.

Evasive Web Bots: As discussed above, in this evaluation phase we consider two

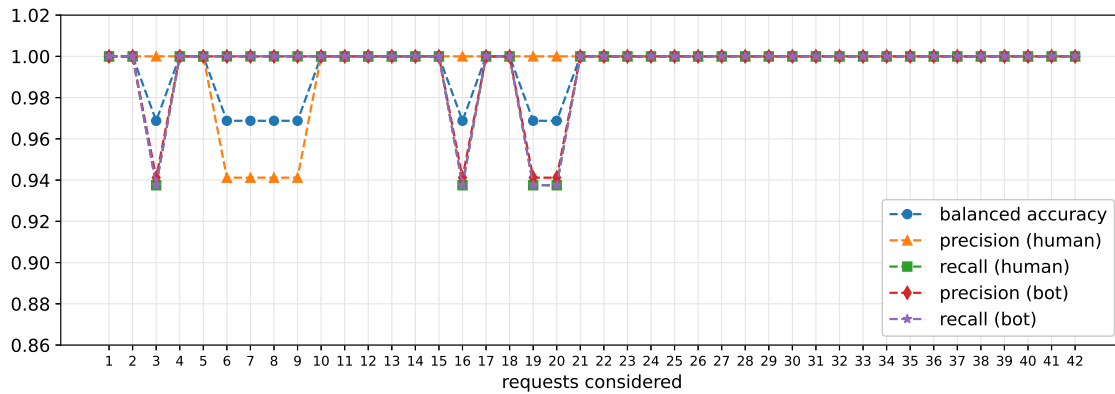


Figure 4.4: Performance of the web bot detection framework

cases: (i) the first case where the web bots do not perform any additional training, and (ii) the second case, where web bots re-train to evade the re-trained web bot detection framework. Table 4.6 presents the results.

Table 4.6: Performance of evasive web bots on the re-trained detection server using the N3 architecture

Requests	Evasion %	Requests	Evasion %
Case #1: no additional training			
300k	2.3%		
Case #2: additional training			
5k	1.1%	50k	1.2%
10k	6.7%	100k	4.9%
15k	1.7%	200k	3.3%
20k	0.5%	300k	5.9%
30k	5.2%	400k	1.2%
40k	2.7%	500k	57.3%

As expected, evading the re-trained web bot detection server is more challenging. We see however that some of the pre-trained web bots from the first evaluation phase manage to evade detection. This can be attributed to the fact that the web server used only 56 bot sessions randomly selected for training which probably did not cover all possible behaviours that the web bots can generate.

Additionally, we see that it is more difficult for web bots to evade the re-trained detection framework. Compared to the first evaluation phase, where web bots managed to achieve a high evasion percentage even from the 30k requests, in the second phase, bots fail to do so until 500k requests. This can be attributed to the fact that the bots used for the training of the detection framework in the first evaluation phase exhibited a more generic behaviour as opposed to the ones from the second (which had to only evade detection). Thus, by having found a behaviour that is closer to humans than the bots from the first evaluation phase (at 500k requests), they were able to start evading detection again and even achieving a higher evasion percentage.

4.2 Evasive Web Bots Using Generative Adversarial Networks

Web bots can generate artificially synthetic humanlike behaviours by utilising real human data. As discussed in Chapter 2, using human data to generate synthetic humanlike data is commonly used in literature to generate evasive behaviours. However, the current approaches use human data primarily to select the parameters of the distributions used to model a humanlike behaviour heuristically.

Thus, in this section, we examine the case of training machine learning models to generate humanlike behaviours using human data. For that, one of the most well-fitted methods that has shown very promising results in the last years is the use of GANs, proposed in Goodfellow et al. (2014). GANs are architectures where two neural networks, the Generator and the Discriminator, contest with each other and are trained simultaneously in an “adversarial” setting. The Generator constantly tries to generate data that can fool the Discriminator into thinking that those data are real (and not synthetically generated). During training, the Generator progressively becomes better at creating data that look real, while the Discriminator becomes better at finding them. As shown in Chapter 3, web bot detection techniques that process and use mouse movements as images have proven to be very effective. Additionally, as shown by Radford et al. (2016), GANs can be used to generate realistic images based on the images they were trained on, which makes them ideal for the generation of humanlike mouse and touchscreen trajectories. Thus, in our setting, GANs are a very promising approach. Even though GANs have been used in Acien et al. (2020b 2021) to generate evasive web bots, they have only been used to generate mobile humanlike touchscreen trajectories and also they process the data as sequences instead of images. Thus, our approach is a novel one and utilises

the capabilities of GANs to generate realistic images. Specifically, in this section we propose a novel method of bots processing mouse trajectories as images and using GANs to generate synthetic humanlike mouse trajectories is proposed. This work was published in Iliou et al. (2021b).

In this section we consider two types of evasive web bots, the (i) scraping web bots that generate humanlike mouse movements while scraping the web pages, and (ii) the mobile web bots that generate humanlike touchscreen events to bypass challenges that require the performance of specific touchscreen events. We follow a common technique for both types of web bots in regards to the generation of the humanlike mouse trajectories and their detection. Specifically, we assume that web bots have already in their possession several human mouse movements and touchscreen events, and can use them to generate new humanlike behaviours. In both cases, the trained detection framework uses as input those images and tries to detect whether those images created from a bot or a human.

Next, we initially present the web bot detection framework used for the evaluation of both types of evasive web bots (Section 4.2.1), and then we present the evasive web bots (Section 4.2.2). After that, we present the experimental setup (Section 4.2.3) and the results of the evaluation of the evasive web bot (Section 4.2.4).

4.2.1 Detection Framework

The web bot detection framework used for the evaluation of the web bots is based on the detection framework presented in Section 3.2.1 that uses mouse trajectories of visitors. Specifically, as shown by Wei et al. (2019) and in Section 3.2, a highly accurate method for detecting web bots based on their mouse movements is to generate images depicting the mouse movements that visitors perform on each web page and feed those into CNNs. A similar approach can be used for mobile touchscreen trajectories, as they can also be processed as images. The general architecture of this web bot detection framework is presented in Figure 4.5.

The framework uses the same approaches presented in Section 3.2.1 for the detection of web bots using mouse trajectories. Specifically, initially images of the mouse movements that each visitor performed on each web page are generated. These images include sequences of the coordinates of all pixels that the mouse passed at a specific time. The data are collected as $\{(x_1, y_1, t_1), \dots, (x_n, y_n, t_n)\}$, where x_i and y_i are the coordinates of the current mouse point, t_i is the respective timestamp, and n is the total

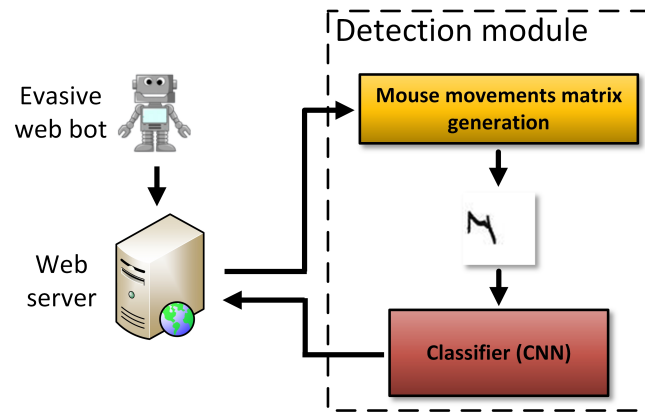


Figure 4.5: Web bot detection framework

number of points over which the mouse hovered. These sequences are mapped into 2-dimensional matrices, where each (x_i, y_i) value pair corresponds to the index of an element in the matrix and $dt_i = t_{i+1} - t_i$ to its value. The same approach is used for touchscreen actions on mobile devices.

The extracted images of the visitors' mouse movements are used as input into a (trained) CNN and the classification result indicates whether the visitor is detected as a bot or a human. In this work, a simple CNN architecture that combines a series of Convolution and Max-pooling layers was used. The architecture is presented in Table 4.7.

Table 4.7: CNN architecture for web bot detection

Layer type	Kernel size / stride	Output Shape	Activation
InputLayer	–	(56, 56, 1)	–
Conv	3x3 / 1	(54, 54, 64)	ReLU
M-Pool	2x2 / 2	(27, 27, 64)	–
Conv	3x3 / 1	(25, 25, 64)	ReLU
M-Pool	2x2 / 2	(12, 12, 64)	–
Flatten	–	(9216)	–
Dense	–	(2)	Softmax

Additionally, images were normalised so that their values become between '0' and '1', a commonly used technique in CNNs. The CNN was implemented using Tensorflow⁴ and the Keras API⁵.

⁴<https://www.tensorflow.org/>

⁵<https://keras.io/>

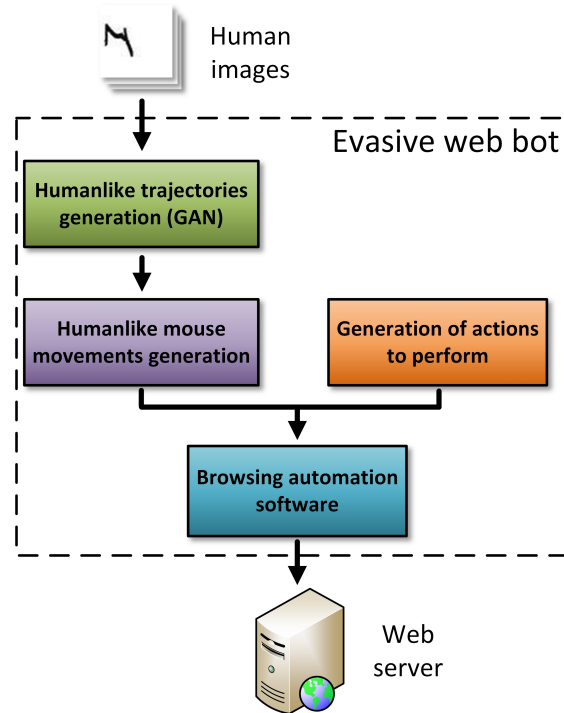


Figure 4.6: Evasive web bots

4.2.2 Evasive Web Bots Using GANs

The web bots evaluated in this chapter use GANs to evade detection based on their mouse trajectories. As shown in Acien et al. (2020b), Wei et al. (2019), Chu et al. (2018), and Section 3.2, detection based on mouse trajectories has shown to be very effective. If the bots can generate humanlike trajectories, then these trajectories can be combined with additional techniques, such as using a browser-like fingerprint and selecting web pages to visit in a humanlike manner as shown in Iliou et al. (2017) and Section 3.2, to enable a more evasive behaviour.

The general architecture of the proposed web bots is presented in Figure 4.6. More specifically, the proposed web bots utilise GANs to generate humanlike mouse movements based on real ones, since, as shown by Radford et al. (2016), GANs have proven to generate realistic images based on the images they were trained on. GANs allow the generation of several mouse trajectories with different starting and ending points each. From the available trajectories, web bots can select the most appropriate ones based on the actions that they want to perform.

GANs are architectures where two neural networks, the Generator and the Discriminator, contest with each other and are trained simultaneously in an “adversarial” setting.

The Generator constantly tries to generate images that can fool the Discriminator into thinking that those images are real. The Generator is trained to take as input points from a latent space (in our case, these points correspond to vectors with values drawn from a Gaussian distribution) and map them into new images. Diversely, the Discriminator is trained to distinguish fake images from real ones, using both images from human trajectories and images created by the Generator. During that process, the Generator becomes better at generating images that look similar to the input images, while the Discriminator becomes better at identifying which images are real.

In this work, a Deep Convolutional Generative Adversarial Network (DCGAN) architecture is used for the generation of humanlike mouse and touchscreen trajectories following the recommendations from Radford et al. (2016). Differentiating from Radford et al. (2016), we used LeakyReLU for both the Generator and the Discriminator, instead of only the Discriminator, because in our preliminary experiments it generated slightly better images. The architecture of the Generator is presented in Table 4.8 while the architecture of the Discriminator is presented in Table 4.9. For the GAN implementation, the Tensorflow⁶ and the Keras API⁷ were used.

4.2.3 Evaluation

To assess the effectiveness of the proposed approach, two types of web bots are considered: (i) the scraping web bots that crawl web servers to harvest their content, and (ii) the mobile web bots that are requested to perform a specific task to prove that they are humans, which in our case is to generate touchscreen trajectories of specific numbers.

Evaluation Methodology

To examine the evasiveness of web bots that utilise GANs for the generation of humanlike mouse movements and touchscreen trajectories, we assume that web bots have already in their possession several human mouse movements and touchscreen events. Bots can then use these data as input to GANs to generate new humanlike behaviours with similar characteristics as the ones they have previously seen.

Additionally, for simplicity, we consider that the web server also uses GANs to train its detection models and that both GANs (the GAN used by the web bots and the GAN used by the web server) have the same architecture and configurations (as presented in

⁶<https://www.tensorflow.org/>

⁷<https://keras.io/>

Table 4.8: Generator Architecture

Layer type	Configs	Output Shape
InputLayer	–	(100)
Dense	–	(12544)
BatchNormalisation	–	(12544)
LeakyReLU	alpha=0.3	(12544)
Reshape	–	(7, 7, 256)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(7, 7, 128)
BatchNormalisation	–	(7, 7, 128)
LeakyReLU	alpha=0.3	(7, 7, 128)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(14, 14, 64)
BatchNormalisation	–	(14, 14, 64)
LeakyReLU	alpha=0.3	(14, 14, 64)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(28, 28, 32)
BatchNormalisation	–	(28, 28, 32)
LeakyReLU	alpha=0.3	(28, 28, 32)
Conv2DTranspose	kernel=5x5, stride=1, padding=same	(56, 56, 1)

Table 4.8 and Table 4.9) and were trained for 20k epochs. We argue that these similarities in the aforementioned methods make the evasion process more difficult.

On the other hand, we consider that the human behaviours used by the web bots to train their models should be different from the ones used by the web server. This choice was made to present a more realistic scenario, where the web server should be faced with new, unseen behaviours. Thus, in our experiments, different human images are used by the web bot detection module and by the evasive web bots.

To evaluate how well the web bots can evade detection, we initially train and evaluate the performance of the web bot detection framework. For that, the considered data sets were split into 80% for the training and 20% for the testing, and the CNN was trained for 30 epochs. Then, this detection framework was used for evaluating the evasive web bots.

Finally, to account for the fact that the performance of both the detection module and the web bots might be affected by the images selected as the training set in each case, we repeated the experiments considering different combinations of the sets to be used

Table 4.9: Discriminator Architecture

Layer type	Configs	Output Shape
InputLayer	–	(56, 56, 1)
Conv	kernel=5x5, stride=2, padding=same	(28, 28, 64)
LeakyReLU	alpha=0.3	(28, 28, 64)
Dropout	rate=0.2	(28, 28, 64)
Conv	kernel=5x5, stride=2, padding=same	(14, 14, 128)
LeakyReLU	alpha=0.3	(14, 14, 128)
Dropout	rate=0.2	(14, 14, 128)
Flatten	–	(25088)
Dense	–	(1)

by the web server and the web bots.

Datasets

The evasive web bots were evaluated on two datasets: (i) the *Web dataset* used in Section 3.2 that was generated by humans while browsing a web server which hosted content copied from Wikipedia⁸, and (ii) part of the *HuMldb dataset*⁹, in which humans were requested to “draw” digits on mobile devices, used in Acien et al. (2020a b).

Web: For this dataset, 27 human subjects were requested to browse a web server that hosted 110 Wikipedia pages from 11 categories/topics. Each human subject was instructed to create two sessions, each session being between 15-20 minutes. All sessions were anonymised and only information that indicates which sessions belong to the same user has been kept.

HuMldb: The second dataset is part of the HuMldb dataset, a dataset that contains a wide range of mobile sensors values acquired during a natural human-mobile interaction performed by more than 600 users; this database is not public but available upon request. To generate this dataset, users were requested to perform eight simple tasks between one and five times (indicating different sessions). In this work, we utilised data from the task where a user had to draw with their finger the digits ‘0’ to ‘9’ over the touchscreen.

The total number of users, sessions, and images for each dataset are presented

⁸<https://www.wikipedia.org/>

⁹<https://github.com/BiDALab/HuMldb>

Table 4.10: Users, Sessions, and images for each dataset

		Set 1	Set 2	Set 3	Total
Web	<i>Users</i>	9	9	9	27
	<i>Sessions</i>	18	18	18	54
	<i>Images</i>	367	369	561	1,297
HuMldb	<i>Users</i>	200	200	200	600
	<i>Sessions</i>	839	847	798	2,484
	<i>Images</i>	8390	8470	7980	24,840

in Table 4.10. In both datasets, and to account for the fact that different visitors have varying monitor resolutions, we re-scaled all images to the same dimensions with a lower resolution. For that, we initially increased the size of each mouse move by 30, i.e., for each pixel where a mouse move was performed (and thus has a non-zero value), its neighbours within distance less or equal to 30 pixels were given the value of that pixel. Then we re-scaled the images to 56x56 dimensions using the Pillow¹⁰ library and with the “antialias” (high-quality downsampling filter) configuration. The 56x56 dimensions were selected to account also for the fact that high resolution images consume a lot of memory when used for training the networks.

Moreover, each dataset was split into three sets to facilitate the evaluation process (see Table 4.10): two were used by the web detection server for training, and one was used by the evasive web bots to generate a humanlike behaviour. To account for the fact that different images of the same user should not be in different sets, the split was performed on a per user basis.

Finally, to account for the randomness introduced when the models are trained on GPU, the experiments were run five times and the average of all runs was considered.

Evaluation Metrics

To assess the effectiveness of web detection, we used the *balanced accuracy* evaluation metric, which is used in the web bot detection problem when the datasets are unbalanced. To evaluate the evasiveness of the web bots and since we have only one class (i.e., only the bots class), we used *recall*, i.e., the percentage of the web bots that were correctly

¹⁰<https://pillow.readthedocs.io/en/stable/>

identified (True Positive) divided by the total number of web bots used (True Positive + False Negatives).

4.2.4 Results

Web Bot Detection Performance

The performance of the detection framework is presented in Table 4.11. Experiments are performed on *set X* & *set Y* (denoted as *X* & *Y*), where a random 80% of *X* and a random 80% of *Y* are used for training the CNN detection framework, and the remaining 20% of *X* and *Y* are used for testing. Data from *set X* are considered as the ‘human’ class, while *set Y* is used as input to the GAN of the bot detection; this GAN generates the same number of images as the number of images in its input.

Table 4.11: Performance of the detection framework

	Balanced accuracy						
X & Y	1 & 2	2 & 1	2 & 3	3 & 2	1 & 3	3 & 1	Avg
Web	0.986	1.000	0.970	0.988	0.974	0.994	0.985
HuMldb	0.995	0.996	0.995	0.997	0.995	0.997	0.996
	Recall						
Web	1.000	1.000	0.996	1.000	1.000	1.000	0.999
HuMldb	0.992	0.996	0.996	0.997	0.995	0.997	0.995

The detection framework manages to achieve a very high accuracy and recall in both datasets. This performance was expected, since, as shown in Wei et al. (2019) and Section 3.2, web bot detection frameworks that use mouse movements have shown very good results. Since the same GAN was used to generate (different) images for training and for testing, the CNN was able to identify this behaviour.

Web Bot Detection Against Evasive Web Bots

To evaluate the proposed web bots, the already trained detection framework that takes advantage of CNNs was used. The evasive web bots train their GAN by using different human images from the ones used by the detection framework. The results are presented in Table 4.12 where the *X* & *Y* indicates the sets that were used for training the web bot detection framework (i.e., Set X and Set Y), and the set *Z* indicates the set that was

used for training the evasive web bots. In this case, the recall indicates the percentage of images that were correctly identified as images generated by web bots divided by the total generated images.



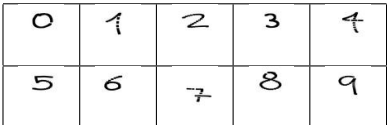
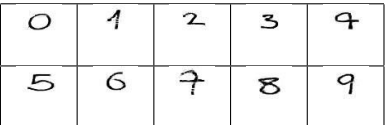
Table 4.12: Performance of evasive web bots

	Recall						
X & Y	1 & 2	2 & 1	2 & 3	3 & 2	1 & 3	3 & 1	Avg
Z	3		1		2		
Web	0.531	0.648	0.693	0.437	0.263	0.139	0.452
HuMldb	0.950	0.943	0.925	0.928	0.950	0.928	0.937

The drop in performance of the detection module when tested on the evasive web bots shows that generating humanlike trajectories using GANs is very effective against web bot detection techniques, even if the same architecture and configurations are used by the detection module. This was expected, since the web server used the same GAN to generate images for its training and validation, but was evaluated with images generated from a different GAN used by the web bots. Additionally, the drop in performance is very high in the case of the *Web dataset* as opposed to the *HuMldb*. This is also something to be expected, since mouse trajectories on a web server vary more between different humans compared to “drawing” numbers on smartphones, making the “modeling” of this behaviour more difficult.

Finally, to qualitatively evaluate the effectiveness of the use of GANs for the generation of humanlike trajectories, selected images are presented in Table 4.13. We observe that web bots have a tendency to select simpler mouse movements to follow instead of more complex ones. This could be attributed to the fact that it is difficult for the latter to be modelled.

Table 4.13: Selected images from humans and generated by GANs

Humans images	Images from humans	Images generated by GANs
Web dataset		
HuMldb		

5 Discussion and Reflection

5.1 Discussion

There is a huge incentive for individuals and companies alike to create web bots that can bypass web bot detection techniques, since such web bots can be used for several malicious purposes. This resulted in a continuous race between researchers on web bot detection techniques and malicious users that try to evade those techniques.

Web servers can use and combine different bot detection methods, both signature-based (e.g., browser fingerprinting), and behaviour-based. Behaviour-based detection techniques can use and combine different machine learning algorithms, by examining information about the web pages bots visit and the mouse movements that they perform. At the same time, the advances in browsing automation enabled the malicious users to effortlessly create evasive advanced web bots that have a browser-like fingerprint, which, enhanced with a humanlike behaviour, can make their detection even harder. Concerning their behaviour, web bots can also use machine learning based techniques to increase the evasiveness.

The results of this research outlined the importance of considering web bots of different sophistication levels (i.e., simple vs moderate vs advanced) when evaluating the web bot detection techniques. The behaviour and characteristics of web bots of different sophistication levels can vary. Additionally, since generating advanced web bots requires a considerable amount of time and effort (and thus malicious actors would probably use them for higher impact attacks), focus should be given on those.

However, we show that several state-of-the-art approaches proposed in literature perform inadequately when faced with advanced web bots. This was not evident in those works most probably because of the lower number of advanced web bots as opposed to simple web bots that masked the ability or lack thereof to detect them. To address this, in this thesis we propose a novel web bot detection framework that combines web logs and mouse movements generated by visitors. This approach results in a very high performance against web bots of different sophistication levels.

Additionally, in this research we show that there are very few works in literature that

examine techniques which can be used by advanced web bots to evade detection based on their behaviour. Thus, we propose two novel evasive web bots that use machine learning techniques to evade detection, (i) one in which web bots use RL to learn an evasive behaviour, and (ii) one in which web bots use GANs to train on human data and generate new synthetic humanlike data. We show that such techniques can be used by web bots to evade detection and that they should be considered by defenders when proposing detection techniques.

The direction and outcome of this thesis goes hand in hand with the literature review. During this thesis we outlined the limitations of web bot detection techniques that use web logs and examined the use of mouse movements, while at the same time other researchers have also started exploring alternatives that also use mouse movements. We went one step further and combined both approaches showcasing that the traditional web bot detection techniques that use web logs can increase the total performance when combined with the ones that use mouse movements. Also, in accordance with relative research on evasive web bots, we show that advanced bots can generate humanlike behaviours that can evade detection. As opposed to current research focusing primarily on using statistics and heuristics to generate such behaviour, we show two novel ways in which the recent advances in machine learning can also be used for that purpose.

The outcomes of this research outline the need for continuous updates of the methods and techniques used by web bot detection frameworks to follow the latest advances of the malicious evasive web bots. As we show, state-of-the-art web bot detection approaches may not be very effective against advanced malicious web bots. This means that, if the detection frameworks do not follow the latest advances of malicious web bots, then such bots may be able to perform several malicious actions that can have a high impact on our society, since there are a lot of services that are accessed via the web. For example, allowing web bots to perform malicious actions on web services may lead to denying service to an actual visitor or to allowing bots unauthorised access to those services and the respective data that they host. This can have devastating results for society. Thus, it is important for detection frameworks to be updated continuously and to use advanced and novel detection techniques (including those proposed in this work) to protect web servers from malicious bots.

Even though novel techniques for both web bot detection and evasion have been proposed in this thesis, the performance trade-offs limit such techniques, especially in the case of the detection techniques. In a real-world web server, where millions of requests

are performed every second, storing all those requests and users' mouse movements and processing them would require a huge amount of resources. Additionally, for the generation of evasive web bots, an attacker would either require a huge amount of IPs that the attacker can "burn" in order to identify an evasive behaviour, or data from several humans which will also require a lot of effort to generate.

Thus, there is a trade-off in both the attacker and defender in regards to the effectiveness of their approaches and the effort required. Depending on the motivation of each party (i.e., web bot creator and web server with web bot detection framework) more weight shall be given to the effectiveness or the efficiency of the approaches.

5.2 Evaluation of Aims and Objectives

In this subsection we reiterate the aims and the objectives of this research, and examine how well they have been fulfilled during this research.

The main research question of this research is to identify how (malicious) advanced web bots can be detected (**R**). To answer this question, this research has three aims: (i) to define advanced web bots by outlining the web bots' landscape including categorising web bots based on their functionality, and sophistication (**A1**), (ii) to evaluate the current and propose novel behaviour-based web bot detection techniques focusing on advanced web bots (**A2**), and (iii) to evaluate how well those detection techniques perform when faced with web bots that use recent advanced in machine learning to evade detection based on their behaviour (**A3**).

To fulfill the aforementioned aims, we initially presented the web bot landscape outlining and categorising the different types of web bots based on their functionality and sophistication (**O1.1**, **O1.2**). This information was used to define the advanced web bots that this research focused on.

Then, we showed the performance limitations of current state-of-the-art approaches when faced with advanced web bots. For that, we used a real-world dataset collected from a public web server (**O2.1**), and evaluated the most prominent (at that time) web bot detection techniques against simple and advanced web bots (**O2.2**). We showed that even though those techniques were very effective against simple web bots, their effectiveness considerably reduced when faced with advanced web bots. This motivated us to propose a novel web bot detection framework that combines both web logs and visitors' mouse movements for the effective detection of advanced web bots (**O2.3**). For

the latter, we created a dataset using several human subjects because no such dataset was available online **(O2.1)**.

Finally, in the last part of this research, we evaluated how well the aforementioned web bot detection techniques perform against advanced web bots that use recent advances in machine learning to evade detection. For that, we initially identified the most fitted machine learning algorithms that can be used by web bots to evade detection **(O3.1)**. Then, we evaluated the aforementioned web bot detection techniques against the web bots that use those machine learning algorithms to evade detection **(O3.2)**.

In conclusion, in this research we:

- Outlined the web bots' landscape, categorising the web bots based on their functionality, and sophistication **(A1)**
- Evaluated the current web bot detection techniques and proposed novel web bot detection techniques for advanced web bots **(A2)**
- Evaluated how these detection techniques perform when faced with advanced web bots that use recent machine learning algorithms to evade detection **(A3)**

Table 5.1: Mapping of the objectives to publications and chapters of this thesis

Objectives	Chapters	Publications
O1.1, O1.2	2	-
O2.1, O2.2	3	Iliou, C. , Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, Y., 2019. Towards a framework for detecting advanced web bots. Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019, ACM, 18:1–18:10.
O2.3	3	Iliou, C. , Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2021. Detection of advanced web bots by combining web logs with mouse behavioural biometrics. Digital Threats: Research and Practice, 2 (3).
O3.1, O3.2	4	<p>Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2021. Web bot detection evasion using generative adversarial networks. IEEE International Conference on Cyber Security and Resilience, CSR 2021, Rhodes, Greece, July 26-28, 2021, IEEE, 115–120.</p> <p>Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., Web bot detection evasion using deep reinforcement learning. Proceedings of the 17th International Conference on Availability, Reliability and Security, ARES 2022, Vienna, Austria, August 23-26, 2022, ACM.</p>

6 Conclusions and Future Work

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. Some of these tasks require web bots to perform highly complex actions, including specific mouse movements, clicks, and keystrokes. Latest advances in browsing automation allow web bots to support the majority of web browsers' functionalities, have a browser-like fingerprint, and perform such highly complex actions. The combination of a browser-like fingerprint and a behaviour that can be configured to be humanlike led malicious actors to use these web bots for malicious purposes, since they can perform complex tasks and, at the same time, avoid detection.

This research aimed to identify techniques that can be used to detect malicious advanced web bots. For that, we had to define the different types of web bots based on their functionality and sophistication, and research techniques that can be used to detect web bots based on their behaviour and investigate how effective those are against advanced web bots that use sophisticated techniques to evade detection. We showed that, even though state-of-the-art web bot detection techniques achieved a very high performance in detecting web bots in general, they performed poorly when faced with only advanced web bots. However, we show that we were able to detect such advanced web bots by comprising two different bot detection modules, and more specifically, (i) a detection module that utilises web logs, and (ii) a detection module that leverages mouse movements. Additionally, we show that malicious web bots can use recent advances in machine learning to evade detection. Specifically, we show that advanced web bots can use (i) RL to evade detection by updating their browsing behaviour based on, among others, whether they have been detected or not, and (ii) GANs to generate (synthetic) images of mouse trajectories similar to those of humans that can be used for browsing.

In summary, in this research we showed that the web bot detection problem is a dynamic one, where both the defenders (i.e., web servers that use web bot detection technologies) as well as the attackers (i.e., the evasive web bots) can continuously update their techniques and methods to achieve their goals. This is very important since it indicates that web bot detection frameworks should be constantly updated to follow the

latest advances of malicious web bots. Otherwise, web bots (especially advanced ones) can bypass the detection techniques and perform malicious actions to web servers which may have devastating results.

In this work, we only examined two machine learning based methods that web bots can use to evade detection. Also, we have proposed a novel way that detection frameworks can use to detect advanced web bots, which, as discussed above, should be continuously updated with new techniques, to follow future updates of malicious web bots. Thus, the future direction of this research is to examine additional techniques that web bots can use to evade detection, such as adversarial machine learning based techniques, and how the existing web bot detection techniques can be enhanced to defend against those. Additionally, new detection techniques tailored to the evasive techniques used by web bots can be investigated. Also, since detection frameworks can use a combination of detection methods, a future direction could be to investigate how different detection methods could be combined together to increase their effectiveness against malicious web bots. Finally, efficiency and optimisation techniques for the current web bot detection approaches proposed in this research should be examined, which will make those techniques more efficient, so as to allow them to be integrated to web servers with limited resources. To summarise, as it is shown from this research, the web bot detection problem is a dynamic one requiring continuous research and updates to make sure that detection frameworks are ready to face malicious web bots and protect the web servers.

Bibliography

- Acien, A., Morales, A., Fierrez, J., Vera-Rodriguez, R. and Bartolome, I., 2020a. Becaptcha: Detecting human behavior in smartphone interaction using multiple inbuilt sensors. *AAAI Workshop on Artificial for Cyber Security (AICS)*.
- Acien, A., Morales, A., Fierrez, J., Vera-Rodríguez, R. and Delgado-Mohatar, O., 2020b. Becaptcha: Bot detection in smartphone interaction using touchscreen biometrics and mobile sensors. *CoRR*, abs/2005.13655. URL <https://arxiv.org/abs/2005.13655>.
- Acien, A., Morales, A., Fierrez, J., Vera-Rodriguez, R. and Delgado-Mohatar, O., 2021. Becaptcha: Behavioral bot detection using touchscreen and mobile sensors benchmarked on humidb. *Engineering Applications of Artificial Intelligence*, 98, 104058.
- von Ahn, L., Blum, M., Hopper, N. J. and Langford, J., 2003. CAPTCHA: using hard AI problems for security. E. Biham, ed., *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, Springer, volume 2656 of *Lecture Notes in Computer Science*, 294–311. URL https://doi.org/10.1007/3-540-39200-9_18.
- Akamai, 2021. Akamai's bot manager - advanced strategies to flexibly manage the long-term business and its impact of bots. URL <https://www.akamai.com/us/en/multimedia/documents/product-brief/bot-manager-product-brief.pdf>.
- Akrout, I., Feriani, A. and Akrouf, M., 2019. Hacking google recaptcha v3 using reinforcement learning. *CoRR*, abs/1903.01003. URL <http://arxiv.org/abs/1903.01003>.
- Alam, S., Dobbie, G., Koh, Y. S. and Riddle, P., 2014. Web bots detection using particle swarm optimization based clustering. *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*, IEEE, 2955–2962. URL <https://doi.org/10.1109/CEC.2014.6900644>.
- Almahmoud, S., Hammo, B. and Al-Shboul, B., 2019. Exploring non-human traffic in online digital advertisements: Analysis and prediction. N. T. Nguyen, R. Chbeir, E. Exposito, P. Aniorté and B. Trawinski, eds., *Computational Collective Intelligence - 11th*

- International Conference, ICCCI 2019, Hendaye, France, September 4-6, 2019, Proceedings, Part II*, Springer, volume 11684 of *Lecture Notes in Computer Science*, 663–675. URL https://doi.org/10.1007/978-3-030-28374-2_57.
- Alqahtani, F. H. and Alsulaiman, F. A., 2020. Is image-based CAPTCHA secure against attacks based on machine learning? an experimental study. *Comput. Secur.*, 88. URL <https://doi.org/10.1016/j.cose.2019.101635>.
- Awad, M. A. and Khalil, I., 2012. Prediction of user's web-browsing behavior: Application of markov model. *IEEE Trans. Syst. Man Cybern. Part B*, 42 (4), 1131–1142. URL <https://doi.org/10.1109/TSMCB.2012.2187441>.
- Azad, B. A., Starov, O., Laperdrix, P. and Nikiforakis, N., 2020. Web runner 2049: Evaluating third-party anti-bot services. C. Maurice, L. Bilge, G. Stringhini and N. Neves, eds., *Detection of Intrusions and Malware, and Vulnerability Assessment - 17th International Conference, DIMVA 2020, Lisbon, Portugal, June 24-26, 2020, Proceedings*, Springer, volume 12223 of *Lecture Notes in Computer Science*, 135–159. URL https://doi.org/10.1007/978-3-030-52683-2_7.
- Bai, Q., Xiong, G., Zhao, Y. and He, L., 2014. Analysis and detection of bogus behavior in web crawler measurement. F. Aleskerov, Y. Shi and A. Lepskiy, eds., *Proceedings of the Second International Conference on Information Technology and Quantitative Management, ITQM 2014, National Research University Higher School of Economics (HSE), Moscow, Russia, June 3-5, 2014*, Elsevier, volume 31 of *Procedia Computer Science*, 1084–1091. URL <https://doi.org/10.1016/j.procs.2014.05.363>.
- Bhargav, A. and Bhargav, M., 2014. Pattern discovery and users classification through web usage mining. *Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014 International Conference on*, IEEE, 632–636.
- Bianco, D., 2013. The pyramid of pain. *Enterprise Detection & Response*. URL <http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html>.
- Bock, K., Patel, D., Hughey, G. and Levin, D., 2017. uncaptcha: A low-resource defeat of recaptcha's audio challenge. W. Enck and C. Mulliner, eds., *11th USENIX Workshop on Offensive Technologies, WOOT 2017, Vancouver, BC, Canada, August 14-15, 2017*, USENIX Association. URL <https://www.usenix.org/conference/woot17/workshop-program/presentation/bock>.

- Buehrer, G., Stokes, J. W., Chellapilla, K. and Platt, J. C., 2009. Classification of automated search traffic. I. King and R. Baeza-Yates, eds., *Weaving Services and People on the World Wide Web, [features some of the cutting-edge research work that were presented at the Workshop Track of the 17th International World Wide Web Conference (WWW 2008) held at Beijing, China, from April 21-25, 2008]*, Springer, 3–26. URL https://doi.org/10.1007/978-3-642-00570-1_1.
- Cabri, A., Suchacka, G., Rovetta, S. and Masulli, F., 2018. Online web bot detection using a sequential classification approach. *20th IEEE International Conference on High Performance Computing and Communications; 16th IEEE International Conference on Smart City; 4th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2018, Exeter, United Kingdom, June 28-30, 2018*, IEEE, 1536–1540. URL <https://doi.org/10.1109/HPCC/SmartCity/DSS.2018.00252>.
- Campobasso, M., Burda, P. and Allodi, L., 2019. CARONTE: crawling adversarial resources over non-trusted, high-profile environments. *2019 IEEE European Symposium on Security and Privacy Workshops, EuroS&P Workshops 2019, Stockholm, Sweden, June 17-19, 2019*, IEEE, 433–442. URL <https://doi.org/10.1109/EuroSPW.2019.00055>.
- Chen, H., He, H. and Starr, A., 2020. An overview of web robots detection techniques. *2020 International Conference on Cyber Security and Protection of Digital Services, Cyber Security 2020, Dublin, Ireland, June 15-19, 2020*, IEEE, 1–6. URL <https://doi.org/10.1109/CyberSecurity49315.2020.9138856>.
- Chen, J., Luo, X., Guo, Y., Zhang, Y. and Gong, D., 2017. A survey on breaking technique of text-based CAPTCHA. *Secur. Commun. Networks*, 2017, 6898617:1–6898617:15. URL <https://doi.org/10.1155/2017/6898617>.
- Chu, Z., Gianvecchio, S. and Wang, H., 2018. Bot or human? A behavior-based online bot detection system. *From Database to Cyber Security - Essays Dedicated to Sushil Jajodia on the Occasion of His 70th Birthday*, 432–449. URL https://doi.org/10.1007/978-3-030-04834-1_21.
- Cloudflare, 2021. Cloudflare bot management. URL <https://www.cloudflare.com/en-gb/products/bot-management/>.
- Datadome, 2020. recaptcha v2 vs v3: are they really efficient for bot protection? URL

<https://datadome.co/bot-detection/recaptchav2-recaptchav3-efficient-bot-protection/>.

DataDome, 2021a. Datadome real-time AI-powered online fraud & bot protection. URL <https://datadome.co/bot-protection-online-fraud-prevention/>.

DataDome, 2021b. How facebook was used as a proxy by web scraping bots. URL <https://datadome.co/bot-detection/how-facebook-was-used-as-a-proxy-by-web-scraping-bots/>.

David, B., DeLong, M. and Filiol, E., 2021. Detection of crawler traps: formalization and implementation - defeating protection on internet and on the TOR network. *J. Comput. Virol. Hacking Tech.*, 17 (3), 185–198. URL <https://doi.org/10.1007/s11416-021-00380-4>.

Distil Networks, 2017. 2017 bad bot report.

Distil Networks, 2018. 2018 bad bot report: The year bad bots went mainstream. URL <https://resources.distilnetworks.com/white-paper-reports/2018-bad-bot-report>.

Distil Networks, 2019. 2019 bad bot report: The bot arms race continues. URL <https://resources.distilnetworks.com/white-paper-reports/bad-bot-report-2019>.

Doran, D. and Gokhale, S. S., 2012. A classification framework for web robots. *J. Assoc. Inf. Sci. Technol.*, 63 (12), 2549–2554. URL <https://doi.org/10.1002/asi.22741>.

Doran, D. and Gokhale, S. S., 2016. An integrated method for real time and offline web robot detection. *Expert Syst. J. Knowl. Eng.*, 33 (6), 592–606. URL <https://doi.org/10.1111/exsy.12184>.

Fawcett, T., 2006. An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27 (8), 861–874. URL <https://doi.org/10.1016/j.patrec.2005.10.010>.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Grzanic, T., Mrcic, L. and Saban, J., 2015. Lino - an intelligent system for detecting malicious web-robots. N. T. Nguyen, B. Trawinski and R. Kosala, eds., *Intelligent Information and Database Systems - 7th Asian Conference, ACIIDS 2015, Bali, Indonesia*,

- March 23-25, 2015, *Proceedings, Part II*, Springer, volume 9012 of *Lecture Notes in Computer Science*, 559–568. URL https://doi.org/10.1007/978-3-319-15705-4_54.
- Haidar, R. and Elbassuoni, S., 2017. Website navigation behavior analysis for bot detection. *2017 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2017, Tokyo, Japan, October 19-21, 2017*, IEEE, 60–68. URL <https://doi.org/10.1109/DSAA.2017.13>.
- Hamidzadeh, J., Zabihimayvan, M. and Sadeghi, R., 2018. Detection of web site visitors based on fuzzy rough sets. *Soft Comput.*, 22 (7), 2175–2188. URL <https://doi.org/10.1007/s00500-016-2476-4>.
- Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2021a. Detection of advanced web bots by combining web logs with mouse behavioural biometrics. *Digital Threats: Research and Practice*, 2 (3). URL <https://doi.org/10.1145/3447815>.
- Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2021b. Web bot detection evasion using generative adversarial networks. *IEEE International Conference on Cyber Security and Resilience, CSR 2021, Rhodes, Greece, July 26-28, 2021*, IEEE, 115–120. URL <https://doi.org/10.1109/CSR51186.2021.9527915>.
- Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, I., 2022. Web bot detection evasion using deep reinforcement learning. *ARES 2022: The 17th International Conference on Availability, Reliability and Security, Vienna, Austria, August 23 - 26, 2022*, ACM, 15:1–15:10. URL <https://doi.org/10.1145/3538969.3538994>.
- Iliou, C., Kostoulas, T., Tsikrika, T., Katos, V., Vrochidis, S. and Kompatsiaris, Y., 2019. Towards a framework for detecting advanced web bots. *Proceedings of the 14th International Conference on Availability, Reliability and Security, ARES 2019, Canterbury, UK, August 26-29, 2019*, ACM, 18:1–18:10. URL <https://doi.org/10.1145/3339252.3339267>.
- Iliou, C., Tsikrika, T., Vrochidis, S. and Kompatsiaris, Y., 2017. Evasive focused crawling by exploiting human browsing behaviour: a study on terrorism-related content. *Proceedings of the 1st International Workshop on Cyber Deviance Detection co-located*

with the Tenth International Conference on Web Search and Data Mining CyberDD @ WSDM 2017), Cambridge, UK, February, 10, 2017.

Imperva, 2019. How bots affect airlines. URL <https://www.imperva.com/resources/reports/How-Bots-Affect-Airlines-.pdf>.

Imperva, 2021. Bad bot report 2021: The pandemic of the internet. URL <https://www.imperva.com/blog/bad-bot-report-2021-the-pandemic-of-the-internet/>.

Jacob, G., Kirda, E., Kruegel, C. and Vigna, G., 2012. PUBCRAWL: protecting users and businesses from crawlers. T. Kohno, ed., *Proceedings of the 21th USENIX Security Symposium, Bellevue, WA, USA, August 8-10, 2012*, USENIX Association, 507–522. URL <https://www.usenix.org/conference/usenixsecurity12/technical-sessions/presentation/jacob>.

Jiang, N. and Dogan, H., 2018. TAPCHA: An Invisible CAPTCHA Scheme. *Proceedings of the 32nd International BCS Human Computer Interaction Conference*, BCS.

Johannes, O., 2016. Principal component analysis (pca) for feature selection and some of its pitfalls. URL http://jotterbach.github.io/2016/03/24/Principal_Component_Analysis/.

Jonker, H., Krumnow, B. and Vlot, G., 2019. Fingerprint surface-based detection of web bot detectors. K. Sako, S. A. Schneider and P. Y. A. Ryan, eds., *Computer Security - ESORICS 2019 - 24th European Symposium on Research in Computer Security, Luxembourg, September 23-27, 2019, Proceedings, Part II*, Springer, volume 11736 of *Lecture Notes in Computer Science*, 586–605. URL https://doi.org/10.1007/978-3-030-29962-0_28.

Kaelbling, L. P., Littman, M. L. and Moore, A. W., 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4, 237–285.

Kwon, S., Kim, Y. and Cha, S. D., 2012. Web robot detection based on pattern-matching technique. *J. Inf. Sci.*, 38 (2), 118–126. URL <https://doi.org/10.1177/0165551511435969>.

Lagopoulos, A. and Tsoumakas, G., 2020. Content-aware web robot detection. *Appl. Intell.*, 50 (11), 4017–4028. URL <https://doi.org/10.1007/s10489-020-01754-9>.

- Laperdrix, P., Bielova, N., Baudry, B. and Avoine, G., 2020. Browser fingerprinting: A survey. *ACM Trans. Web*, 14 (2), 8:1–8:33. URL <https://doi.org/10.1145/3386040>.
- Loyola-González, O., Monroy, R., Medina-Pérez, M. A., Cervantes, B. and Grimaldo-Tijerina, J. E., 2018. An approach based on contrast patterns for bot detection on web log files. I. Z. Batyrshin, M. de Lourdes Martínez-Villaseñor and H. E. P. Espinosa, eds., *Advances in Soft Computing - 17th Mexican International Conference on Artificial Intelligence, MICAI 2018, Guadalajara, Mexico, October 22-27, 2018, Proceedings, Part I*, Springer, volume 11288 of *Lecture Notes in Computer Science*, 276–285. URL https://doi.org/10.1007/978-3-030-04491-6_21.
- Marenzi, O., 2019. Web scraping for investments. Technical report, Opimas. URL <http://www.opimas.com/research/436/detail/>.
- Mikolov, T., Chen, K., Corrado, G. and Dean, J., 2013. Efficient estimation of word representations in vector space. Y. Bengio and Y. LeCun, eds., *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. URL <http://arxiv.org/abs/1301.3781>.
- Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. A., 2013. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602. URL <http://arxiv.org/abs/1312.5602>.
- Molloy, D., 2021. The great graphics card shortage of 2020 (and 2021). URL <https://www.bbc.com/news/technology-55755820>.
- Na, D., Park, N., Ji, S. and Kim, J., 2020. Captchas are still in danger: An efficient scheme to bypass adversarial captchas. I. You, ed., *Information Security Applications - 21st International Conference, WISA 2020, Jeju Island, South Korea, August 26-28, 2020, Revised Selected Papers*, Springer, volume 12583 of *Lecture Notes in Computer Science*, 31–44. URL https://doi.org/10.1007/978-3-030-65299-9_3.
- Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F. and Vigna, G., 2013. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. *2013 IEEE Symposium on Security and Privacy, SP 2013, Berkeley, CA, USA, May 19-22, 2013*, IEEE Computer Society, 541–555. URL <https://doi.org/10.1109/SP.2013.43>.

- Oikonomou, G. and Mirkovic, J., 2009. Modeling human behavior for defense against flash-crowd attacks. *Proceedings of IEEE International Conference on Communications, ICC 2009, Dresden, Germany, 14-18 June 2009*, IEEE, 1–6. URL <https://doi.org/10.1109/ICC.2009.5199191>.
- Park, K., Pai, V. S., Lee, K. and Calo, S. B., 2006. Securing web service by automatic robot detection. A. Adya and E. M. Nahum, eds., *Proceedings of the 2006 USENIX Annual Technical Conference, Boston, MA, USA, May 30 - June 3, 2006*, USENIX, 255–260. URL <http://www.usenix.org/events/usenix06/tech/park.html>.
- Pastrana, S., Thomas, D. R., Hutchings, A. and Clayton, R., 2018. Crimebb: Enabling cybercrime research on underground forums at scale. *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018*, 1845–1854. URL <https://doi.org/10.1145/3178876.3186178>.
- PerimeterX, 2021. Perimeterx bot defender. URL <https://www.perimeterx.com/downloads/product-briefs/PerimeterX-Product-Brief-Bot-Defender.pdf>.
- Pudil, P., Novovicová, J. and Kittler, J., 1994. Floating search methods in feature selection. *Pattern Recognition Letters*, 15 (10), 1119–1125. URL [https://doi.org/10.1016/0167-8655\(94\)90127-9](https://doi.org/10.1016/0167-8655(94)90127-9).
- Radford, A., Metz, L. and Chintala, S., 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. Y. Bengio and Y. LeCun, eds., *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. URL <http://arxiv.org/abs/1511.06434>.
- Rahman, R. U. and Tomar, D. S., 2021. Threats of price scraping on e-commerce websites: attack model and its detection using neural network. *J. Comput. Virol. Hacking Tech.*, 17 (1), 75–89. URL <https://doi.org/10.1007/s11416-020-00368-6>.
- Rovetta, S., Cabri, A., Masulli, F. and Suchacka, G., 2017. Bot or not? A case study on bot recognition from web session logs. A. Esposito, M. Faúndez-Zanuy, F. C. Morabito and E. Pasero, eds., *Quantifying and Processing Biomedical and Behavioral Signals*, Springer, volume 103 of *Smart Innovation, Systems and Technologies*, 197–206. URL https://doi.org/10.1007/978-3-319-95095-2_19.

- Rovetta, S., Suchacka, G. and Masulli, F., 2020. Bot recognition in a web store: An approach based on unsupervised learning. *J. Netw. Comput. Appl.*, 157, 102577. URL <https://doi.org/10.1016/j.jnca.2020.102577>.
- Rude, N. and Doran, D., 2015. Request type prediction for web robot and internet of things traffic. T. Li, L. A. Kurgan, V. Palade, R. Goebel, A. Holzinger, K. Verspoor and M. A. Wani, eds., *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*, IEEE, 995–1000. URL <https://doi.org/10.1109/ICMLA.2015.53>.
- Rui, C., Jing, Y., Rong-gui, H. and Shu-guang, H., 2013. A novel lstm-rnn decoding algorithm in captcha recognition. *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control*, IEEE, 766–771.
- Schwarz, M., Lackner, F. and Gruss, D., 2019. Javascript template attacks: Automatically inferring host information for targeted exploits. *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*, The Internet Society.
- Seyyar, M. B., Çatak, F. Ö. and Gül, E., 2017. Detection of attack-targeted scans from the apache http server access logs. *Applied Computing and Informatics*.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T. P., Leach, M., Kavukcuoglu, K., Graepel, T. and Hassabis, D., 2016. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529 (7587), 484–489. URL <https://doi.org/10.1038/nature16961>.
- Sisodia, D. S., Verma, S. and Vyas, O. P., 2015. Agglomerative approach for identification and elimination of web robots from web server logs to extract knowledge about actual visitors. *Journal of Data Analysis and Information Processing*, 3 (01), 1.
- Sivakorn, S., Polakis, I. and Keromytis, A. D., 2016. I am robot: (deep) learning to break semantic image captchas. *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, 388–403.
- Stassopoulou, A. and Dikaiakos, M. D., 2009. Web robot detection: A probabilistic rea-

- soning approach. *Comput. Networks*, 53 (3), 265–278. URL <https://doi.org/10.1016/j.comnet.2008.09.021>.
- Stevanovic, D., An, A. and Vlajic, N., 2012. Feature evaluation for web crawler detection with data mining techniques. *Expert Syst. Appl.*, 39 (10), 8707–8717. URL <https://doi.org/10.1016/j.eswa.2012.01.210>.
- Stevanovic, D., Vlajic, N. and An, A., 2013. Detection of malicious and non-malicious website visitors using unsupervised neural network learning. *Appl. Soft Comput.*, 13 (1), 698–708. URL <https://doi.org/10.1016/j.asoc.2012.08.028>.
- Suchacka, G., Cabri, A., Rovetta, S. and Masulli, F., 2021. Efficient on-the-fly web bot detection. *Knowl. Based Syst.*, 223, 107074. URL <https://doi.org/10.1016/j.knsys.2021.107074>.
- Vastel, A., Rudametkin, W., Rouvoy, R. and Blanc, X., 2020. Fp-crawlers: studying the resilience of browser fingerprinting to block crawlers. *MADWeb'20-NDSS Workshop on Measurements, Attacks, and Defenses for the Web*.
- Wang, D., Xi, L., Zhang, H., Liu, H., Zhang, H. and Song, T., 2015a. Web robot detection with semi-supervised learning method. *3rd International Conference on Material, Mechanical and Manufacturing Engineering (IC3ME 2015)*, Atlantis Press, 2123–2128.
- Wang, G., Konolige, T., Wilson, C., Wang, X., Zheng, H. and Zhao, B. Y., 2013. You are how you click: Clickstream analysis for sybil detection. S. T. King, ed., *Proceedings of the 22th USENIX Security Symposium, Washington, DC, USA, August 14-16, 2013*, USENIX Association, 241–256. URL <https://www.usenix.org/conference/usenix-security13/technical-sessions/presentation/wang>.
- Wang, J., Zhang, M., Yang, X., Long, K. and Xu, J., 2015b. Http-scan: detecting http-flooding attack by modeling multi-features of web browsing behavior from noisy web-logs. *China Communications*, 12 (2), 118–128.
- Watson, C. and Zaw, T., 2018. *OWASP Automated Threat Handbook: Web Applications*. OWASP Foundation.
- Wei, A., Zhao, Y. and Cai, Z., 2019. A deep learning approach to web bot detection using mouse behavioral biometrics. *Biometric Recognition - 14th Chinese Conference, CCBR 2019, Zhuzhou, China, October 12-13, 2019, Proceedings*, 388–395. URL https://doi.org/10.1007/978-3-030-31456-9_43.

- Xie, Y. and Yu, S., 2009. A large-scale hidden semi-markov model for anomaly detection on user browsing behaviors. *IEEE/ACM Trans. Netw.*, 17 (1), 54–65. URL <http://doi.acm.org/10.1145/1514070.1514075>.
- Xu, X., Liu, L. and Li, B., 2020. A survey of CAPTCHA technologies to distinguish between human and computer. *Neurocomputing*, 408, 292–307. URL <https://doi.org/10.1016/j.neucom.2019.08.109>.
- Yang, R., Wang, X., Chi, C., Wang, D., He, J., Pang, S. and Lau, W. C., 2021. Scalable detection of promotional website defacements in black hat SEO campaigns. M. Bailey and R. Greenstadt, eds., *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, USENIX Association, 3703–3720. URL <https://www.usenix.org/conference/usenixsecurity21/presentation/yang-ronghai>.
- Yang, Y., Vljajic, N. and Nguyen, U. T., 2015. Next generation of impersonator bots: Mimicking human browsing on previously unvisited sites. *IEEE 2nd International Conference on Cyber Security and Cloud Computing, CSCloud 2015, New York, NY, USA, November 3-5, 2015*, IEEE Computer Society, 356–361. URL <https://doi.org/10.1109/CSCloud.2015.93>.
- Yu, S., Guo, S. and Stojmenovic, I., 2015. Fool me if you can: Mimicking attacks and anti-attacks in cyberspace. *IEEE Trans. Computers*, 64 (1), 139–151. URL <https://doi.org/10.1109/TC.2013.191>.
- Yu, S., Zhao, G., Guo, S., Xiang, Y. and Vasilakos, A. V., 2011. Browsing behavior mimicking attacks on popular web sites for large botnets. *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, IEEE, 947–951.
- Zabihi, M., Jahan, M. V. and Hamidzadeh, J., 2014. A density based clustering approach for web robot detection. *2014 4th International Conference on Computer and Knowledge Engineering (ICCKE)*, IEEE, 23–28.
- Zabihimayvan, M. and Doran, D., 2018. Some (non-)universal features of web robot traffic. *52nd Annual Conference on Information Sciences and Systems, CISS 2018, Princeton, NJ, USA, March 21-23, 2018*, IEEE, 1–6. URL <https://doi.org/10.1109/CISS.2018.8362266>.
- Zabihimayvan, M., Sadeghi, R., Rude, H. N. and Doran, D., 2017. A soft computing

approach for benign and malicious web robot detection. *Expert Syst. Appl.*, 87, 129–140. URL <https://doi.org/10.1016/j.eswa.2017.06.004>.

Zeiler, M. D., Taylor, G. W. and Fergus, R., 2011. Adaptive deconvolutional networks for mid and high level feature learning. D. N. Metaxas, L. Quan, A. Sanfeliu and L. V. Gool, eds., *IEEE International Conference on Computer Vision, ICCV 2011, Barcelona, Spain, November 6-13, 2011*, IEEE Computer Society, 2018–2025. URL <https://doi.org/10.1109/ICCV.2011.6126474>.

Appendix A - Features Proposed in Literature for Web Logs

In this section we present the features that are extracted from web logs by different works in literature to train machine learning models.

Category	Feature	Literature
Total requests	Total HTTP requests	Park et al. (2006), Stassopoulou and Dikaiakos (2009), Buehrer et al. (2009), Stevanovic et al. (2012 2013), Alam et al. (2014), Sisodia et al. (2015), Grzinic et al. (2015), Hamidzadeh et al. (2018), Rovetta et al. (2017 2020)
	Number/Percentage of requests with HTTP request method GET, HEAD, POST, etc. Usually several features are created here, each one for each request method	Park et al. (2006), Stevanovic et al. (2012 2013), Sisodia et al. (2015), Grzinic et al. (2015), Hamidzadeh et al. (2018), Zabihimayvan et al. (2017), Rovetta et al. (2017 2020), Cabri et al. (2018), Suchacka et al. (2021)
	Number/Percentage of requests with HTTP response code 2xx, 3xx, 4xx. Usually several features are created here, each one for each response code type (e.g., one for all responses with code 2xx)	Park et al. (2006), Stassopoulou and Dikaiakos (2009), Stevanovic et al. (2012 2013), Zabihi et al. (2014), Sisodia et al. (2015), Hamidzadeh et al. (2018), Zabihimayvan et al. (2017), Rovetta et al. (2017 2020), Cabri et al. (2018), Suchacka et al. (2021), Rahman and Tomar (2021)

	Total Bytes downloaded	Stevanovic et al. (2013), Sisodia et al. (2015), Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017), Cabri et al. (2018), Rovetta et al. (2020), Suchacka et al. (2021), Rahman and Tomar (2021)
	Standard deviation of the Bytes of the pages downloaded	Alam et al. (2014), Rahman and Tomar (2021)
	Total Bytes uploaded	Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017)
File types	Number/Percentage of requests of specific type such as JavaScript, CSS, images, etc. Usually several features are created here, each one for each file type	Park et al. (2006), Stassopoulou and Dikaiakos (2009), Stevanovic et al. (2012 2013), Sisodia et al. (2015), Doran and Gokhale (2016), Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017), Haidar and Elbassuoni (2017), Cabri et al. (2018), Suchacka et al. (2021)
	Number/Percentage of requests requests to favicon.ico	Park et al. (2006)
	Max number/Percentage of requests requests to embedded objects in web page	Park et al. (2006), Zabihi et al. (2014), Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017), Cabri et al. (2018)
	HTML-to-image ratio	Stevanovic et al. (2012 2013), Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017), Rovetta et al. (2017)
	image-to-page ratio	Rovetta et al. (2020)

Referrer field	Number/Percentage of requests with empty referrer	Stevanovic et al. (2012 2013), Sisodia et al. (2015), Hamidzadeh et al. (2018), Rovetta et al. (2017 2020), Zabihimayvan et al. (2017), Cabri et al. (2018), Suchacka et al. (2021)
	Number/Percentage of requests with referrer	Park et al. (2006)
	Percentage of requests with unknown/unseen referrer	Park et al. (2006)
URL depth	Standard deviation of depth	Stevanovic et al. (2012 2013), Hamidzadeh et al. (2018), Zabihimayvan et al. (2017)
	Maximum depth	Hamidzadeh et al. (2018), Zabihimayvan et al. (2017)
	Width of HTTP request paths	Hamidzadeh et al. (2018)
	Maximum depth of HTTP request paths	Sisodia et al. (2015)
URL patterns	Percentage of consecutive sequential HTTP requests	Stevanovic et al. (2012 2013), Hamidzadeh et al. (2018), Zabihimayvan et al. (2017)
	Number of repeated requests	Sisodia et al. (2015)
	Penalty value for back-and-forward navigation or loop	Zabihi et al. (2014), Hamidzadeh et al. (2018), Zabihimayvan et al. (2017)
Time	Session duration (seconds)	Stassopoulou and Dikaiakos (2009), Alam et al. (2014), Sisodia et al. (2015), Grzinic et al. (2015), Hamidzadeh et al. (2018), Zabihimayvan et al. (2017), Rovetta et al. (2017 2020), Suchacka et al. (2021), Rahman and Tomar (2021)
	Session duration (categorical: small/medium/long)	Almahmoud et al. (2019)

	Browsing speed / request rate	Buehrer et al. (2009)
	Average time between successive requests	Sisodia et al. (2015), Haidar and Elbassuoni (2017)
	Standard deviation of inter-request times	Grzinic et al. (2015), Zabihimayvan et al. (2017), Cabri et al. (2018), Rahman and Tomar (2021)
	Mean time spend on each page	Rovetta et al. (2017 2020)
	Entropy of inter-request times	Buehrer et al. (2009), Haidar and Elbassuoni (2017), Rahman and Tomar (2021)
	Time taken to serve requests	Hamidzadeh et al. (2018)
Specific pages visited	Number of visits to <i>home</i> web page	Rovetta et al. (2017 2020)
	Vector showing which pages are visited	Haidar and Elbassuoni (2017)
	Vector showing the order of visited pages	Haidar and Elbassuoni (2017)
	Vector showing how many times each web page was visited	Haidar and Elbassuoni (2017)
	Vector showing the time spend on each web page	Haidar and Elbassuoni (2017)
	Average requests par web page	Haidar and Elbassuoni (2017)
Other generic features	Number/Percentage of trap file or hidden links requests, or binary value indicating if it was accessed	Zabihi et al. (2014), Grzinic et al. (2015), Hamidzadeh et al. (2018), Zabihimayvan et al. (2017)
	Number of <i>robots.txt</i> requests or binary value indicating if it was accessed	Stassopoulou and Dikaiakos (2009), Stevanovic et al. (2012), Sisodia et al. (2015), Grzinic et al. (2015)

	Number/Percentage of night-time requests	Sisodia et al. (2015), Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017)
	Binary indicating if multiple IPs have been used	Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017)
	Binary indicating if multiple agent names have been used	Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017)
	Binary indicating if session ID has been changed	Grzinic et al. (2015)
	Average value of page popularity index of visited web pages	Stevanovic et al. (2013), Hamidzadeh et al. (2018), Zabihi-mayvan et al. (2017)
	IP reputation	Buehrer et al. (2009)
	Clicks performed in session	Almahmoud et al. (2019)
	Number of IPs/cities generated from the same user ID	Buehrer et al. (2009)
Domain specific features (e.g., e-commerce, search engines)	Whether the session ended with a purchase	Rovetta et al. (2017 2020)
	Whether the session was performed by the site administrator	Rovetta et al. (2017)
	Number of login operations	Rovetta et al. (2017 2020)
	Number of views of the page with shipping terms and conditions	Rovetta et al. (2017 2020)
	Number of searches using the internal search engine	Rovetta et al. (2017 2020)
	The number of pages of type "Browse"	Rovetta et al. (2017)

Number of views of product description pages	Rovetta et al. (2017 2020)
Number of operations of adding a product to the shopping cart	Rovetta et al. (2017 2020)
Number of views of pages informing about the store and the trading company	Rovetta et al. (2017 2020)
Number of views of pages with entertainment contents	Rovetta et al. (2017 2020)
Number of other page views	Rovetta et al. (2020)
Whether a "source" of the session is specified	Rovetta et al. (2020)
Query associated to a specific category (e.g., have keywords related to adult content, etc.)	Buehrer et al. (2009)
Query that have keywords used usually in spam	Buehrer et al. (2009)
Queries are performed in alphabetical or close to alphabetical order	Buehrer et al. (2009)
Click-to-query ration (for search engines)	Buehrer et al. (2009)
Query keyword entropy	Buehrer et al. (2009)
Query keyword length entropy	Buehrer et al. (2009)
Query keyword length entropy	Buehrer et al. (2009)
Using search operators (such as the <i>inURL</i> and <i>intitle</i>)	Buehrer et al. (2009)
Query category entropy	Buehrer et al. (2009)

Appendix B - Methods and Performance of Literature

In this section we present the performance of the different methods proposed in literature as well as a summary visualisation. S, M, and A correspond to the simple, moderate, and advanced bots (respectively). Each set of algorithms under the same bot type for the same work indicate different experimental setups or datasets used by this work.

Table B.1: Performance of behaviour-based web bot detection techniques that use web logs. Red color is used to highlight very low performances

Literature	Type	Algorithm	F-score	Acc.	AUC
Park et al. (2006)	S+M	AdaBoost	–	0.95	–
Stassopoulou and Dikaiakos (2009)	S	Naive Bayes	0.855	–	–
	S	Naive Bayes	0.866	–	–
	S	Naive Bayes	0.903	–	–
	S	Naive Bayes	0.866	–	–
	S	Naive Bayes	0.863	–	–
Buehrer et al. (2009)	S+M	AdaBoost	0.86	0.89	–
		ADTree	0.87	0.90	–
		BaggingTrees	0.91	0.93	–
		Bayesian Net.	0.87	0.90	–
		Logist. Regr.	0.89	0.91	–
		Naive Bayes	0.87	0.90	–
		PART	0.86	0.87	–
	S+M	AdaBoost	–	0.76	–
		BaggingTrees	–	0.81	–
		Bayesian Net.	–	0.78	–
		Logist. Regr.	–	0.83	–
		Naive Bayes	–	0.71	–
		PART	–	0.82	–

		Rand. Forest	–	0.82	–
Stevanovic et al. (2012)	S	C4.5	~0.71	~0.99	–
		RIPPER	~0.73	~0.99	–
		kNN	~0.68	~0.98	–
		Naive Bayes	~0.48	~0.96	–
		Bayesian Net.	~0.67	~0.98	–
		SVM	~0.69	~0.99	–
		MLP	~0.73	~0.99	–
	S	C4.5	~0.73	~0.99	–
		RIPPER	~0.71	~0.99	–
		kNN	~0.76	~0.99	–
		Naive Bayes	~0.44	~0.97	–
		Bayesian Net.	~0.67	~0.99	–
		SVM	~0.67	~0.99	–
		MLP	~0.81	~0.99	–
Sisodia et al. (2015)	S	AdaBoost	~0.58	–	–
		Bagging	~0.59	–	–
		Rand. Forest	~0.59	–	–
		C4.5	~0.58	–	–
		Naive Bayes	~0.24	–	–
Grzinic et al. (2015)	S+M	C4.5	0.972	–	0.773
		SVM	0.979	–	0.801
	S+M	C4.5	0.992	–	0.985
		SVM	0.997	–	0.978
Doran and Gokhale (2016)	S	DTMC	~0.73	–	–
	S	DTMC	~0.89	–	–
	S	DTMC	~0.91	–	–
Rovetta et al. (2017)	S	MLP	0.94	0.96	–
		SVM	0.98	0.98	–
		k-means	0.98	0.98	–
		GPCM	0.97	0.98	–
Haidar and Elbassuoni (2017)	S	SVM	0.770	0.790	0.852
		MLP	0.774	0.793	0.876

		Naive Bayes	0.780	0.799	0.882
		Rand. Forest	0.770	0.800	0.880
		AdaBoost	0.780	0.801	0.884
	S	SVM	0.760	0.789	0.841
		MLP	0.773	0.791	0.862
		Naive Bayes	0.788	0.796	0.879
		Rand. Forest	0.756	0.797	0.878
		AdaBoost	0.780	0.800	0.881
Cabri et al. (2018)	S	MLP	~0.98	~0.98	–
Loyola-González et al. (2018)	S+M	kNN	–	–	0.9988
		Bagging	–	–	0.9864
		Bayesian Net.	–	–	0.9999
		C4.5	–	–	0.9989
		Logist. Regr.	–	–	timeout
		MLP	–	–	timeout
		Naive Bayes	–	–	0.9997
		Rand. Forest	–	–	1.0000
		SVM	–	–	timeout
		Miner (RF)	–	–	0.9773
		Miner (Bagg.)	–	–	0.9366
		LCMiner	–	–	0.9624
	S+M	kNN	–	–	0.9988
		Bagging	–	–	timeout
		Bayesian Net.	–	–	0.9999
		C4.5	–	–	0.9955
		Logist. Regr.	–	–	timeout
		MLP	–	–	timeout
		Naive Bayes	–	–	0.9999
		Rand. Forest	–	–	1.0000
		SVM	–	–	timeout
		Miner (RF)	–	–	0.9851
		Miner (Bagg.)	–	–	0.9588
		LCMiner	–	–	0.9652

Suchacka et al. (2021)	S	NN-based	0.96	0.97	–
		DTMC	0.89	0.89	–
	S	NN-based	0.96	0.96	–
		DTMC	0.78	0.82	–
Rahman and Tomar (2021)	S	kNN	~0.79	~0.79	–
		Naive Bayes	~0.86	~0.84	–
		Decis. Tree	~0.86	~0.85	–
		SVM	~0.86	~0.85	–
		Cascade NN	~0.92	~0.91	–
		Rand. Forest	~0.86	~0.87	–
		Logist. Regr.	~0.86	~0.86	–
Almahmoud et al. (2019)	S+M	SVM	0.886	–	–
		kNN	0.976	–	–
		AdaBoost	0.824	–	–
		Bagging (RF)	0.958	–	–
		Decis. Tree	0.941	–	–
Rovetta et al. (2020)	S	MLP	0.985	0.986	–
		SVM	0.991	0.992	–
		k-means	0.991	0.992	–
		GPCM	0.985	0.986	–
			Precision		
Alam et al. (2014)	S	HPSO	0.97		
			Jaccard	RI	
Zabihi et al. (2014)	S	DBSCAN	0.940	0.997	
	S	DBSCAN	0.951	0.9805	
Hamidzadeh et al. (2018)	S	SOM+FRS	0.96	–	
		SOM+ART2	0.84	–	
		DBSCAN	0.90	–	
	S	SOM+FRS	0.81	–	
		SOM+ART2	0.73	–	
		DBSCAN	0.76	–	
	S	SOM+FRS	0.88	–	
SOM+ART2		0.79	–		

		DBSCAN	0.84	–
Zabihimayvan et al. (2017)	S	FRS+MCL	0.8918	0.9319
		SOM+ART2	0.8747	0.9193
		DBSCAN	0.5718	0.5718
	S	FRS+MCL	0.4412	0.9103
		SOM+ART2	0.3191	0.8891
		DBSCAN	0.2248	0.8675

Table B.2: Performance of behaviour-based web bot detection techniques that use mouse trajectories of visitors

Literature	Type	Algorithm	Prec.	Rec.	F-score	Acc.	AUC
Chu et al. (2018)	M	C4.5	–	0.9964	–	–	0.9999
	M ¹	C4.5	–	0.9660	–	–	0.9997
	M+A	C4.5	–	0.9945	–	–	0.9996
Wei et al. (2019)	M	GBDT	–	0.9934	–	–	0.9925
		RNN	–	0.9340	–	–	0.9740
		CNNs	–	0.9981	–	–	0.9984
	M	GBDT	–	–	–	–	0.0630
		RNN	–	–	–	–	0.1720
		CNNs	–	–	–	–	0.9240
Acien et al. (2021)	M+A	SVM	0.881	0.829	0.854	0.858	0.936
		kNN	0.823	0.780	0.800	0.801	0.900
		Rand. Forest	0.977	0.968	0.973	0.965	0.997
		OneClassSVM	–	–	–	0.571	–
	M+A	SVM	0.988	0.996	0.992	0.992	0.992
		kNN	0.985	0.992	0.989	0.989	0.991
		Rand. Forest	1.000	0.997	0.998	0.998	0.999
		OneClassSVM	–	–	–	0.805	–
	M+A	SVM	0.740	0.999	0.850	0.824	0.936
		kNN	0.782	0.999	0.877	0.860	0.877
		Rand. Forest	0.780	0.999	0.876	0.858	0.924
		GANs	0.899	0.912	0.908	0.888	0.934
	M+A	SVM	0.620	0.988	0.762	0.688	0.886
		kNN	0.600	0.986	0.647	0.601	0.812
		Rand. Forest	0.532	0.998	0.669	0.544	0.992
		GANs	0.773	0.755	0.764	0.744	0.811

¹Bots that repeat human actions that are used as input to them, but do not exhibit any intelligence.

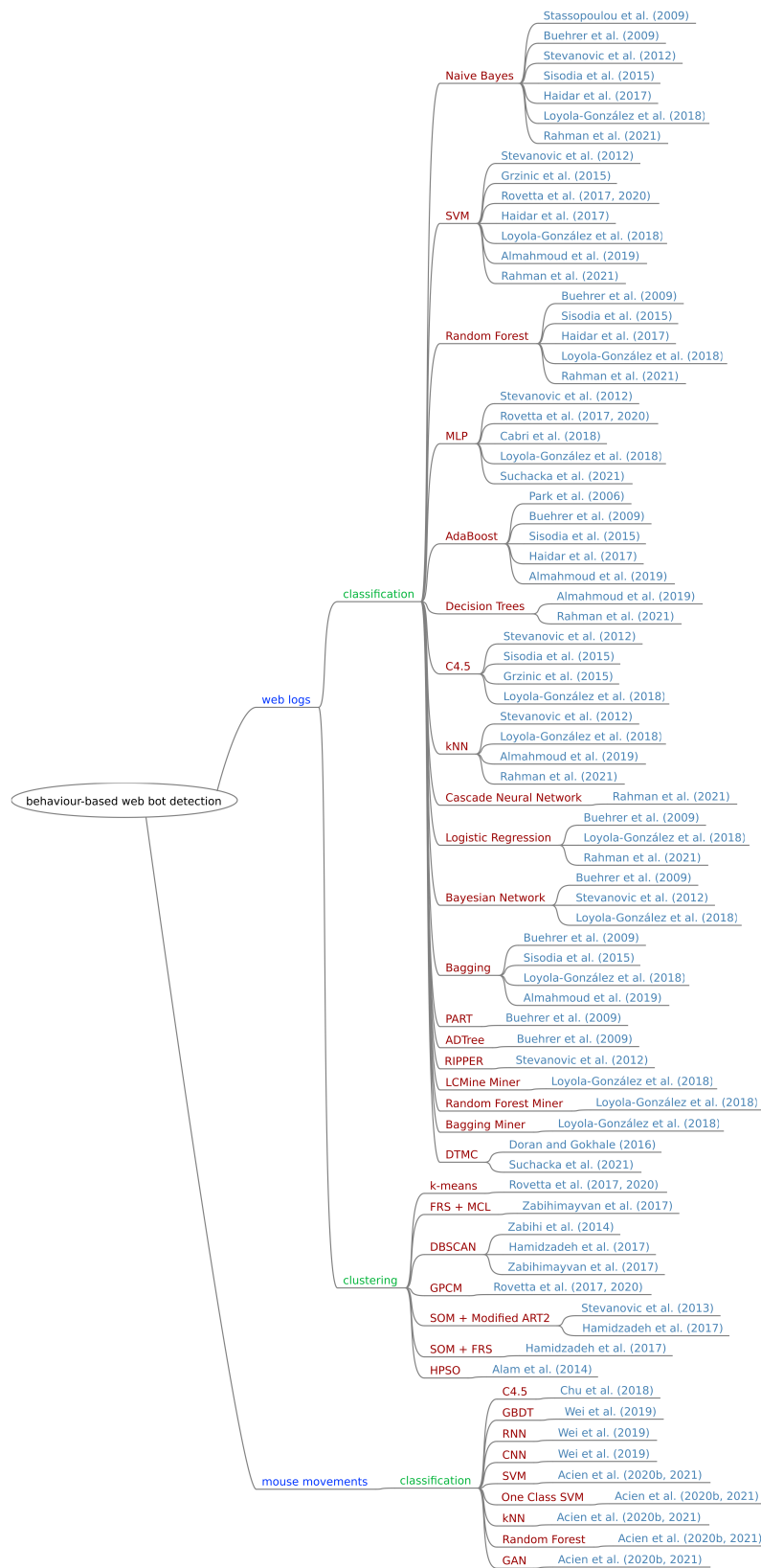


Figure B.1: Classification of articles based on web bot sophistication and the detection method followed

Appendix C - Features for Web Logs Used

In this Table we present the features that extracted from the web logs and used by our proposed web bot detection frameworks.

Id	Feature	Short description
1	Total requests	Total number of HTTP requests that the agent issued during the session.
2	Total session Bytes	The sum of all requested pages' size (in Bytes) in a session.
3	HTTP GET requests	Total number of HTTP GET requests issued during the session.
4	HTTP POST requests	Total number of HTTP POST requests issued during the session.
5	HTTP HEAD requests	Total number of HTTP HEAD requests issued during the session.
6	% HTTP 3xx requests	The percentage of HTTP requests that led to an HTTP 3xx code response.
7	% HTTP 4xx requests	The percentage of HTTP requests that led to an HTTP 4xx code response.
8	% image requests	The percentage of HTTP requests that requested an image. This feature searches for all known image formats' ending.
9	% PDF requests	The percentage of HTTP requests that requested a PDF file.
10	% CSS file requests	The percentage of HTTP requests that requested a CSS file.
11	% JS requests	The percentage of HTTP requests that requested a JavaScript file.

12	HTML-to-image ratio	The number of the requested HTML files divided by the number of requested image files in a session.
13	% requests with unsigned referrers	The percentage of total HTTP requests that had no refer.
14	Search engine refer	Binary. If a session has at least one request with a known search engine refer.
15	Unknown refer	Binary. Refer exists, but not from the aforementioned search engines.
16	Depth SD	Standard deviation of requested pages' depth (i.e. number of '/' in URL path).
17	Max requests per page	The maximum number of requests to the same page in a session.
18	Average requests per page	The average number of requests per page in a session.
19	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	% of consecutive sequential HTTP requests	The percentage of HTTP requested URLs that contain the previously requested URL as a subpart.
21	Session time	The total time (in seconds) between the first and the last HTTP request of the session.
22	Browsing speed	The ratio of the total number of requested pages over time (in seconds).
23	SD of inter-request times	Standard deviation of time between successive requests.

Appendix D - Ethics

In this Appendix, the following files related to the collection of data from human subjects used in this research are attached:

- The ethics checklist for the collection of the data
- The participant information sheet given to the human subjects
- The participant agreement form that each human subject had to sign before participating to the experiment

The risk of the part of this research that used human subjects was found to be *low*.

About Your Checklist	
Ethics ID	22296
Date Created	12/09/2018 21:47:52
Status	Approved
Date Approved	19/10/2019 09:09:45
Date Submitted	15/10/2019 17:11:48
Risk	Low

Researcher Details	
Name	Christos Iliou
Faculty	Faculty of Science & Technology
Status	Postgraduate Research (MRes, MPhil, PhD, DProf, EngD, EdD)
Course	Postgraduate Research - FST
Have you received funding to support this research project?	No
Please list any persons or institutions that you will be conducting joint research with, both internal to BU as well as external collaborators.	Bournemouth University (BU) and Centre for Research & Technology Hellas (CERTH) [https://www.certh.gr/root.en.aspx]

Project Details	
Title	Advanced web bots: Machine learning based detection and evasion techniques
Start Date of Project	10/02/2017
End Date of Project	10/02/2022
Proposed Start Date of Data Collection	16/10/2019
Original Supervisor	Vasilis Katos
Approver	Marcin Budka
Summary - no more than 600 words (including detail on background methodology, sample, outcomes, etc.)	
<p>Automated programs (bots) are responsible for a large percentage of website traffic. These bots can be used for malicious purposes including, but not limited to, content scraping, vulnerability scanning, account takeover, distributed denial of service attacks, marketing fraud, carding and spam. Thus, web servers must be equipped with the tools to detect such web bots.</p> <p>The purpose of this research is to create a web bot detection framework. To achieve that, we are planning to monitor human and bot interactions on a testbed web server. During these interactions, several software and hardware characteristics of visitor's machine as well as behavioural characteristics of visitors will be recorded and stored. We will use these interactions to create models to distinguish human visitors from web bots.</p>	

Filter Question: Does your study involve Human Participants?

Participants	
Describe the number of participants and specify any inclusion/exclusion criteria to be used	
We are looking for as many participants as possible. There are no special requirements for the selected participants. Any adult having access to a computer and the Internet is eligible to participate.	
Do your participants include minors (under 16)?	No
Are your participants considered adults who are competent to give consent but considered vulnerable?	No
Is a Disclosure and Barring Service (DBS) check required for the research activity?	No

Recruitment	
Please provide details on intended recruitment methods, include copies of any advertisements.	
We will recruit students from the Department of Computing and Informatics at Bournemouth University by word of mouth.	
Do you need a Gatekeeper to access your participants?	No

Data Collection Activity	
Will the research involve questionnaire/online survey? If yes, don't forget to attach a copy of the questionnaire/survey or sample of questions.	No
Will the research involve interviews? If Yes, don't forget to attach a copy of the interview questions or sample of questions	
Will the research involve a focus group? If yes, don't forget to attach a copy of the focus group questions or sample of questions.	No
Will the research involve the collection of audio materials?	No
Will your research involve the collection of photographic materials?	No
Will your research involve the collection of video materials/film?	No
Will the study involve discussions of sensitive topics (e.g. sexual activity, drug use, criminal activity)?	No
Will any drugs, placebos or other substances (e.g. food substances, vitamins) be administered to the participants?	No
Will the study involve invasive, intrusive or potential harmful procedures of any kind?	No
Could your research induce psychological stress or anxiety, cause harm or have negative consequences for the participants or researchers (beyond the risks encountered in normal life)?	No
Will your research involve prolonged or repetitive testing?	No

Consent	
Describe the process that you will be using to obtain valid consent for participation in the research activities. If consent is not to be obtained explain why.	
We will give the participants a copy of the Participant information Sheet and the Participant Agreement Form prior to the experiments	

allowing them to read it carefully. We will be available to answer any questions for any potential participants.	
Do your participants include adults who lack/may lack capacity to give consent (at any point in the study)?	No
Will it be necessary for participants to take part in your study without their knowledge and consent?	No

Participant Withdrawal	
At what point and how will it be possible for participants to exercise their rights to withdraw from the study?	
Participants can withdraw during the process of the experiment (i.e. browsing the website) at any time and without giving a reason. To be able to remove their data, participants have to tell us the PHP session id that their browser had during the experiments. For those who does not know how to find it, we will assist them.	
If a participant withdraws from the study, what will be done with their data?	
If a participant decides to withdraw, we will remove any data collected about him/her. Once the data has been collected and processed participants can still withdraw their data up to the point where the data is analysed and incorporated into the research findings or outputs. Withdrawing their data at this point may also adversely affect the validity and integrity of the research.	

Participant Compensation	
Will participants receive financial compensation (or course credits) for their participation?	No
Will financial or other inducements (other than reasonable expenses) be offered to participants?	No

Research Data	
Will identifiable personal information be collected, i.e. at an individualised level in a form that identifies or could enable identification of the participant?	No
Will research outputs include any identifiable personal information i.e. data at an individualised level in a form which identifies or could enable identification of the individual?	No

Storage, Access and Disposal of Research Data	
Where will your research data be stored and who will have access during and after the study has finished.	
Once your project completes, will any anonymised research data be stored on BU's Online Research Data Repository "BORDaR"?	Yes

Dissemination Plans	
Will you inform participants of the results?	

Final Review	
Are there any other ethical considerations relating to your project which have not been covered above?	No

Risk Assessment	
Have you undertaken an appropriate Risk Assessment?	Yes

Attached documents

PIS_v0.5.docx - attached on 09/10/2019 17:13:07
PAF_v0.5.docx - attached on 09/10/2019 17:13:10
PAF_v0.6.docx - attached on 19/06/2020 10:43:18

Approved Amendments	
Message	I would like to update the PAF.docx by replacing the statement "I understand that my data may be used to support other research projects in the future, including future publications, reports, or presentations." with the statement "I understand that my data may be used to support other research projects in the future, including future publications, reports, or presentations. No identifiable information will be included on these data." To clarify that also in this case no identifiable information will be included. I am attaching the respective document.
Date Submitted	19/06/2020 10:43
Comment	
Date Approved	29/06/2020 19:28
Approved By	Marcin Budka



Participant Information Sheet

The title of the research project

Advanced web bots: Machine learning based detection and evasion techniques

Invitation to take part

You are being invited to take part in a research project. Before you decide it is important for you to understand why the research is being done and what it will involve. Please take time to read the following information carefully and discuss it with others if you wish. Ask us if there is anything that is not clear or if you would like more information. Take time to decide whether or not you wish to take part.

Who is organising/funding the research?

This research is a joined initiative between the Bournemouth University (BU - <https://www.bournemouth.ac.uk/>) and the Centre for Research & Technology Hellas (CERTH - <https://www.certh.gr/root.en.aspx>). The persons responsible are professor Vasilis Katos and Dr. Theodoros Kostoulas from BU, Dr. Ioannis Kompatsiaris and Dr. Stefanos Vrochidis from CERTH and Mr. Christos Iliou from both BU and CERTH.

What is the purpose of the project?

Automated programs (bots) are responsible for a large percentage of website traffic. These bots can be used for malicious purposes including, but not limited to, content scraping, vulnerability scanning, account takeover, distributed denial of service attacks, marketing fraud, carding and spam. Thus, web servers must be equipped with the tools to detect such web bots.

The purpose of this research is to create a web bot detection framework. To achieve that, we are planning to monitor human and bot interactions on a testbed web server. During these interactions, several software and hardware characteristics of visitor's machine as well as behavioural characteristics of visitors will be recorded and stored. Based on these interactions, we are planning to create machine learning based web bot detection models. The outcome of this research will be used as part of Christos Iliou's PhD.

Why have I been chosen?

There are no special requirements for the selected participants. Anyone having access to a computer and the Internet is eligible to participate.

Do I have to take part?

It is up to you to decide whether or not to take part. If you do decide to take part, you will be given this information sheet to keep and be asked to sign a participant agreement form. You can withdraw from participation during the process of the experiment (i.e. browsing the website) at any time and

without giving a reason. If you decide to withdraw, we will remove any data collected about you. To be able to remove your data, you have to send us the PHP session id that your browser had during the experiments. If you do not know how to find that, you can ask us to assist you. Upon request we will remove any collected information related to you within 3 days of your request.

Once the data has been collected and processed you can still withdraw your data up to the point where the data is analysed and incorporated into the research findings or outputs. Withdrawing your data at this point may also adversely affect the validity and integrity of the research. Deciding to take part or not will not impact upon/adversely affect your education or studies at BU (or that of others).

What would taking part involve?

By taking part in this research you will have to visit a website that contains information about specific topics. You will have to browse this website like you would browse any other website. For example, you may move your mouse while reading the text of the website, you may click on any hyperlink that interests you. There are no limits or requirements in respect of how many webpages you will have to read/visit and how much time you will put in each webpage. However, it is highly recommended that you spend some time on the web server and visit several hyperlinks. It would be better if you are behaving in this site like you would behave if you were visiting, for example, wikipedia.org and reading about a topic (and several subtopics) that you are interested in.

What are the advantages and possible disadvantages or risks of taking part?

Whilst there are no immediate benefits for those people participating in the project, it is hoped that the outcome of this research will provide mechanisms for the protection of web servers from web bots. As web bots are an issue which affects most Internet activities, we believe that there is an indirect benefit for everyone who will take part in this research.

To our knowledge, there are no possible disadvantages or risks associated with taking part in this research.

What type of information will be sought from me and why is the collection of this information relevant for achieving the research project's objectives?

We are planning to collect hardware and software information about visitors' devices. To be able to collect the aforementioned information a conventional browser software must be used with JavaScript and cookies enabled. Supported browsers are Firefox, Chrome, Edge and Opera, but we believe that any conventional browser can be used. Please do not use any anonymisation browsers (such as Tor browser), anonymisation software/plugins or proxies because we might not be able to collect all the information we need. If you have any doubt about the browser you are planning to use or its configuration, please feel free to contact us.

Besides the software and hardware specifications of your machine, we will store all your interactions with the testbed web server. This monitoring includes all interactions, both from the keyboard and the mouse, on this web server.

When you finish the experiments, you can simply remove all cookies from your browser that have been downloaded from our web server and all the information stored locally on your machine will be

removed. If you do not know how to do that, you let us know which browser software you used and we will tell you which steps to follow to clear all the cookies created from our web server.

Will I be recorded, and how will the recorded media be used?

The data will be made publicly available. No identifiable information will be included on these data.

How will my information be kept?

All the information we collect about you during the course of the research will be kept strictly in accordance with current data protection legislation. Research is a task that we perform in the public interest, as part of our core function as a university. Bournemouth University (BU) is a Data Controller of your information which means that we are responsible for looking after your information and using it appropriately. BU's Research Participant Privacy Notice sets out more information about how we fulfil our responsibilities as a data controller and about your rights as an individual under the data protection legislation. We ask you to read this Notice¹ so that you can fully understand the basis on which we will process your information.

Publication

You will not be identifiable in any external reports or publications about the research. Research results will be published to reputable conferences and journals and will also be used in the dissertation of Christos Iliou's PhD.

Security and access controls

The collected data will be stored temporarily in a password protected database that is only accessible through a whitelist of IPs. After the experiments are finished, we will export and delete the content of the online database and we will transfer them to an offline database to perform experiments on them. No identifiable information will be stored in the offline database.

Sharing and further use of your personal information

The dataset will be made public and will not include any identifiable information.

Retention of your data

We are planning to make the dataset public. The dataset will not include any identifiable information.

Contact for further information

If you have any questions or would like further information, please contact:

Mr Christos Iliou
Emails: ciliou@bournemouth.ac.uk
iliouchristos@iti.gr

¹ <https://intranetsp.bournemouth.ac.uk/documentsrep/Research%20Participant%20Privacy%20Notice.pdf>

Dr Theodoros Kostoulas
Email: tkostoulas@bournemouth.ac.uk

Professor Vasilis Katos
Email: vkatos@bournemouth.ac.uk

In case of complaints

Any concerns about the study should be directed to Theodoros Kostoulas (tkostoulas@bournemouth.ac.uk) or Panos Amelidis (pamelidis@bournemouth.ac.uk). If you concerns have not been answered, you should contact Professor Tiantian Zhang, Deputy Dean for Research and Professional Practice at the Faculty of Science and Technology, Bournemouth University by email to researchgovernance@bournemouth.ac.uk .

Finally

If you decide to take part, you will be given a copy of the information sheet and a signed participant agreement form to keep.

Thank you for considering taking part in this research project.



Participant Agreement Form

Full title of project: (“the Project”) Advanced web bots: Machine learning based detection and evasion techniques

Name, position and contact details of researcher: Christos Iliou, PhD candidate, iliouchristos@iti.gr or ciliou@bournemouth.ac.uk

Name, position and contact details of supervisor: Vasilis Katos, Professor, vkatos@bournemouth.ac.uk

Agreement to participate in the study

You should only agree to participate in the study if you agree with all of the statements in this table and accept that participating will involve the listed activities.

I have read and understood the Participant Information Sheet (PIS.docx) and have been given access to the BU Research Participant Privacy Notice which sets out how we collect and use personal information (https://www1.bournemouth.ac.uk/about/governance/access-information/data-protection-privacy).	
I have had an opportunity to ask questions.	
I understand that my participation is voluntary. I can stop participating in research activities at any time without giving a reason and I am free to decline to answer any particular question(s).	
I agree that BU researchers may access and/or process information about the device used to access the testbed webserver and my browsing activity on the webserver as described in the Participant Information Sheet.	
I understand that taking part in the research will include the following activity/activities as part of the research:	
<ul style="list-style-type: none">the collection of hardware and software information from my device.	
<ul style="list-style-type: none">being monitored while browsing the webserver. Monitoring includes all interactions on the webserver (both from keyboard and mouse).	
I understand that, if I withdraw from the study, I will also be able to withdraw my data from further use in the study except where it will be harmful to the project to have my data removed.	
I understand that my data will be published and/or archived at BU's Online Research Data Repository. No identifiable information will be included on these data.	
I understand that my data will be made publicly available for research purposes. No identifiable information will be included on these data.	
I understand that my data may be used to support other research projects in the future, including future publications, reports, or presentations. No identifiable information will be included on these data.	
	Initial box to agree
I consent to take part in the project on the basis set out above	

I confirm my agreement to take part in the project on the basis set out above.

Name of participant
(BLOCK CAPITALS)

Date
(dd/mm/yyyy)

Signature

Name of researcher
(BLOCK CAPITALS)

Date
(dd/mm/yyyy)

Signature