Exposé for Bachelor Thesis

How well can privacy preserving machine learning be utilized to implement a decentralized and privacy focused bot detection system for websites?

submitted by

Matz-Jona Radloff

Matriculation number 6946325

Study Program: Computer Science

submitted on February 7, 2022

Supervisor: August See

First reviewer: Prof. Dr. Mathias Fischer

Zweitgutachter: N.N.

Abstract

Malicious use of automated bots present an increasing risk to applications in the web. Existing solutions do either not perform well, are not accessible to many providers due to high cost, or disregard modern privacy standards. This work aims to provide a proof-of-concept for a basic system that incorporates all of the above criterions and compares different combinations of the system with and without federated learning and personal data.

Contents

1 Motivation **4**

2 Related Work **5**
2.1 Proprietary Solutions 5
2.2 Scientific Work 5

3 Method **7**
3.1 Concept 7
3.2 Datasets 7
3.2.1 Human Data 7
3.2.2 Bot Data 8
3.3 Machine Learning Model 8
3.4 Evaluation 9

4 Time plan **11**

Bibliography **12**

1 Motivation

In this work, the term bot is referring to software that is automatically performing HTTP(S) requests with the intent of harming the target or reaching another malicous goal. While this threat is nothing new to the web the attack surface has grown significantly over the past year [Laba; Labb]. Especially the increased usage of web interfaces in poorly secured IoT devices and the trend to (re-)implement software as web applications is responsible for this.

The usage of bots can have several goals. This thesis primarily focuses on detecting web-based bots that try to access or perform actions on websites but other and related attack types exist, for example: DoS attacks aim to overload the target's infrastructure such that it becomes inaccessible for normal use. Carding and Credential stuffing refers to performing payment or login requests to find working credit card numbers and credentials usually obtainend from a data breach. Data scrapers download the website data and can use the data for malicious purposes, e.g. damage SEO or violate copyrights. Content spam includes inserting malicious or polluting data on platforms that allow user generated content. Scalping or inventory hoarding of shopping items can artificially raise prices, damage brands, generate false market forces and create a bad customer experience.

Recent studies show that of 2020's internet traffic 25.6% was fraudulent and automatically generated [Laba] [Labb]. They also show that both the percentage of bot traffic in general as well as malicious bot traffic has increased over time.

Most of the above attacks need to trick the webserver and application backends into performing the request as if it had been initiated by a human. Instead of combating the resulting issues separately, bot detection could potentially mitigate many at once.

A complication in this problem space is the, often desired, requirement for non-malicious bots to be granted normal access. A prominent example are scraper bots used by search engines that need to request websites periodically to build their search indices. A common technique to exploit this requirement is trying to emulate known bot signatures from large search engines, e.g. Googlebot [ADJ18].

Many website operators tend to use solutions that are easy to integrate. This requires embeding external software which collects user data and sends it to servers of the software vendors. Closed source software does not allow to determine what exactly happens to the user data and website operators open themselves to additional threats in case of a data breach. Depending on the operating countries of both the websites and software vendors, data privacy regulation might also not allow sharing user data at all or require the operator to document the concrete data transfer in a very detailed and legally complicated way, e.g. in countries falling under the GDPR [Uni]. Because of the above reasons it is desirable to either employ self-hosted software or use a solution that does not require user data transfers.

2 Related Work

2.1 Proprietary Solutions

https://datadome.co/
https://www.perimeterx.com/products/bot-defender/
https://www.imperva.com/products/advanced-bot-protection-management/
https://www.fastly.com/products/cloud-security/bot-protection
https://www.cloudflare.com/products/bot-management/
https://developers.google.com/recaptcha/docs/v3
https://www.hcaptcha.com/

2.2 Scientific Work

The paper [Li+21] introduces a federated learning approach similar to the goals of this work but differs in the specific use case and implementation. Their system focuses on the detection of IoT (Internet of Things) devices which are easily hacked and turned into zombies. These zombies are commonly used in DDoS (Distributed Denial of Service) attacks which their strategy tries to make not feasible to perform. They also develop their own iterative model averaging based method "gated recurrent unit" (GRU) which is optimized for their specific use case.

Iliou et al. [Ili+19] present a comparison of different machine learning algorithms and combinations of different attributes used in previous literature. The attributes comprise mostly of request metadata which would be suitible for a privacy-friendly bot detection system, for example the percentage of image requests or the number of total bytes per session. The authors split the bot data in their dataset into simple and advanced bots which is determined by whether the request have a browser agent name and, in case it does, whether the IP has shown malicious activity before. Their results show that different sets of attributes are performing best depending on the classification algorithm used. The best performing one is Random Forest although the paper concludes that using an ensemble classifier that averages over all used methods would be more stable. Additionally simple web bots can be detected very easily while detecting advanced bots is significantly harder, with areas under the ROC curve of 1.00 and 0.64 respectively. Especially in false positive intolerant use cases the performance of detecting advanced bots is too poor to be used in the real world. The authors conclude that future work would need to incorporate more advanced features which can not be easily simulated by bots.

The work of [Pap+21] outlines the problems and privacy-realted concerns really well and tries to solve a very similar problem but focuses on mobile devices. The authors run a pre-trained machine learning model on the user's device. To avoid local changes to the model a cryptographic proof is generated that is verified on a server.

Among others, the works of Shen et al. [SCG12] and Antal et al. [AD19] [AE18] show the viability of using mouse and trackpad actions to verify the authenticity of users but privacy concerns often stand in the way of using such a method in practice.

Additionally Acien et al. [Aci+20] show the feasibility of using biometric features for bot detection in general and propose new mouse trajectory synthesis methods as well as a GAN-based learning system that can distinguish between humans and bots with 93% accuracy with only one mouse trajectory as input.

3 Method

3.1 Concept

The thesis explores how additional data, which might be personal user data, can potentially improve a machine learning system for bot detection. It also shows how federated learning can be used to allow using user data while mainting privacy.

For comparison a similar system to the proposed architecture of Iliou et al. [Ili+19] is implemented. The authors propose a machine learning based web bot detection framework which operates on request log data. They also identify which extracted features of the data perform the best in this context. Their method was tested on a year's worth of HTTP log data from MK-Lab's public web server[1]. The data includes IP addresses, request method, request path, referrers, user agent strings and timestamps. Iliou et al.'s work it used as the basis for the machine learning system of this thesis because they compared the most promising features that have been proposed over 5 years prior to their publication (2019) and accumulated their findings in concise results.

Mouse, touchpad or touch gestures, which are not considered by Iliou et al. [Ili+19], supplement the data in this work's approach. This type of data is harder to fake by attackers trying to emulate human behavior and would be classified as sensitive user data. It could not be reasonably used by a third party provider and would require explicit user consent [Uni].

To avoid the complicated privacy-realted implications federated learning, a technique that combines both machine learning and respects modern privacy standards [KMR15] [Kon+16], is used and compared against.

3.2 Datasets

3.2.1 Human Data

To realistically match request metadata and user mouse data a dataset is gathered from two websites in a user experiment. The websites have a basic blog-style structure with different information sections and user registration features. Their web server logs all relevant data to extract the features required for the metadata approach while a javascript library records mouse data and sends it to the site's backends for storage.

While no suitible datasets exist that include both request and mouse data, many publicly available datasets exist that contain valid user mouse movement and click data. If it is going seem usefull, the experiments data could be augmented by these. Shen et al.'s [SCG12] dataset contains mouse dynamics information from 28 users and 30 sessions per users which each contain around 3000 mouse events. The DFL dataset [AD19] includes 20-30 sessions of 21 users. The Balabit

1. Multimedia Knowledge and Social Media Analytics Laboratory, https://mklab.iti.gr/

Mouse Dynamics Challenge Data Set [Bal] includes a few longer session and several shorter session which are meant to be used for training and testing respectively. For the purposes of bot detection, both can be used.

If needed, additional datasets are available, e.g.: [2] [3]

3.2.2 Bot Data

To generate bot request data two different approaches are implemented by using the selenium-python [Pyt] library which runs automated actions in an actual browser environment. Instead of the standard chromedriver which runs a headless google chrome instance, undetected-chromedriver [ult] is used which is a patched version that claims to not trigger many anti-bot services. If time allows it, both variants could be compared against, too.

The first, naive approach just performs the desired actions, e.g. scraping information or registering a user account. The second includes efforts to make the requests seem more human, for example by using randomized delays or requesting different pages before performing the actual target actions.

Mouse data labeled as bot input will be generated using two methods. The first naive approach interpolates linearly between the start and end point of randomly generated movements. The second method uses existing open source libraries (Probably one of [4],[5]). More complicated simulations exist but their implementations are not publicly available. [Hu+17] [Naz03] The two methods used depict a reasonable choice of a basic attack that tries to evade detection by fake mouse movement.

Raw mouse data is segmented into mouse actions such as mouse move (MM) or mouse move and a click (point and click, PC) similar to [AD19] and their previous paper [AE18]. Multiple results can be averaged to increase detection performance.

3.3 Machine Learning Model

Many different machine learning models are suitable for binary classification. Hu et al. [Hu+17] compare different classifiers in a context where mouse data is used. Random Forest and Multilayer Perceptron perform the best. Iliou et al. [Ili+19] also show that Random Forest performs the best when using request metadata. If time allows it, the thesis will compare multiple classification methods against each other and only use one otherwise.

Iliou et al. [Ili+19] ranked the best performing metrics for simple and advanced bots per classification algorithm. The following are listed for Random Forest and advanced bots:

1. Total number of HTTP HEAD requests issued during the session. (5)
2. The percentage of HTTP requests that led to an HTTP 4xx code response. (7)
3. The percentage of HTTP requests that requested a css file. (10)

2. https://figshare.com/articles/dataset/Mouse_Behavior_Data_for_Static_Authentication/5619313
3. https://www.uvic.ca/ecs/ece/isot/datasets/?utm_medium=redirect&utm_source=/engineering/ece/isot/datasets/
4. https://github.com/AntoinePassemiers/Realistic-Mouse
5. https://github.com/patrikoss/pyclick

4. The percentage of HTTP requests that requested a JavaScript file. (11)
5. The number of the requested HTML files divided by the number of requested image files in a session. (12)
6. Boolean indicating if a session has at least one request with a known search engine refer. (14)
7. The percentage of HTTP requested URLs that contain the previously requested URL as a subpart. (20)
8. The total time (in seconds) between the first and the last HTTP request of the session. (21)

Attribute 14 might not be suitible as the users in the experiment will be asked to access the websites directly. The websites are otherwise designed such that all attributes are meaningful.

The mouse data input features will consist of the (probably normalized) x- and y-coordinates as well as a time value for each mouse event. Additional features will be engineered similar to [AE18], such as mean, standard deviation, minimum and maximum value of path tangent, horizontal, vertical and overall velocity, acceleration, jerk, angular velocity. Additionally the type of action, length of the movement and time needed to complete the action will be used.

A system for pre-processing the different datasets will be developed.

The thesis will either use Tensorflow with and without federated learning or scikit-learn with-/without Flower. Tensorflow is one of the most widely used machine learning frameworks and the integration of its runtime into distributed learners in the form of website backends or even client devices is feasible. Scikit-learn and Flower seem to be more accessible but both approaches will be compared. The thesis does not include the actual integration into such systems. A simulated environment of multiple learner instances that have access to a secure communications channel will be developed.

3.4 Evaluation

The performance of the following scenarios will be evaluated and compared:

1. Only request metadata, data from both websites, naive bot data
2. Only request metadata, data from both websites, advanced bot data
3. Request metadata and mouse data, data from both websites, naive bot data
4. Request metadata and mouse data, data from both websites, advanced bot data
5. Only request metadata, combined data by federated learning, naive bot data
6. Only request metadata, combined data by federated learning, advanced bot data
7. Request metadata and mouse data, combined data by federated learning, naive bot data
8. Request metadata and mouse data, combined data by federated learning, advanced bot data

The performance will be primarily quantified by bote detection rate. Additionally the accuracy will be compared by providing the confusion matrices and ROC curve visualizations of all scenarios. Especially the false positive rate will be weighed very highly as bot detection is false positive intolerant to not disturb the user experience.

4 Time plan

My planned steps in roughly two week intervals:

1. Implement the basic websites and plan the experiment
2. Start implementing data aggregator and feature extractor
3. Implement the bot data generator (only naive types)
4. Build the ML model, verify that it works and that it can classify the data correctly
5. Compare the metadata and mouse data approaches
6. Integrate Federated learning and build the simulated environment of multiple distributed learners
7. Extend the bot data generation
8. Formulate and perform the experiments to test the thesis' hypothesis
9. Process and visualize the results

Bibliography

[Aci+20] Alejandro Acien et al. *BeCAPTCHA-Mouse: Synthetic Mouse Trajectories and Improved Bot Detection*. In: *ArXiv* abs/2005.00890 (2020).

[AD19] Margit Antal and Lehel Denes-Fazakas. *User Verification Based on Mouse Dynamics: a Comparison of Public Data Sets*. In: *2019 IEEE 13th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. 2019, pp. 143–148. DOI: 10.1109/SACI46893.2019.9111596.

[ADJ18] Nilani Algiryage, Gihan Dias, and Sanath Jayasena. *Distinguishing Real Web Crawlers from Fakes: Googlebot Example*. In: *2018 Moratuwa Engineering Research Conference (MERCon)*. 2018, pp. 13–18. DOI: 10.1109/MERCon.2018.8421894.

[AE18] Margit Antal and Elod Egyed-Zsigmond. *Intrusion Detection Using Mouse Dynamics*. In: *CoRR* abs/1810.04668 (2018). arXiv: 1810.04668. URL: http://arxiv.org/abs/1810.04668.

[Bal] Balabit. *Releasing the Balabit Mouse Dynamics Challenge Data Set*. URL: https://medium.com/balabit-unsupervised/releasing-the-balabit-mouse-dynamics-challenge-data-set-a15a016fba6c (visited on 12/02/2021).

[Hu+17] Shujie Hu et al. *Deceive Mouse-Dynamics-Based Authentication Model via Movement Simulation*. In: *2017 10th International Symposium on Computational Intelligence and Design (ISCID)*. Vol. 1. 2017, pp. 482–485. DOI: 10.1109/ISCID.2017.134.

[Ili+19] Christos Iliou et al. *Towards a Framework for Detecting Advanced Web Bots*. In: *Proceedings of the 14th International Conference on Availability, Reliability and Security*. ARES '19. Canterbury, CA, United Kingdom: Association for Computing Machinery, 2019. ISBN: 9781450371643. DOI: 10.1145/3339252.3339267. URL: https://doi.org/10.1145/3339252.3339267.

[KMR15] Jakub Konečný, Brendan McMahan, and Daniel Ramage. *Federated Optimization: Distributed Optimization Beyond the Datacenter*. In: *CoRR* abs/1511.03575 (2015). arXiv: 1511.03575. URL: http://arxiv.org/abs/1511.03575.

[Kon+16] Jakub Konečný et al. *Federated Optimization: Distributed Machine Learning for On-Device Intelligence*. In: *CoRR* abs/1610.02527 (2016). arXiv: 1610.02527. URL: http://arxiv.org/abs/1610.02527.

[Laba] Imperva Threat Research Lab. *Bad Bot Report 2020: Bad Bots Strike Back*. URL: https://www.imperva.com/blog/bad-bot-report-2020-bad-bots-strike-back/ (visited on 11/12/2021).

[Labb] Imperva Threat Research Lab. *Bad Bot Report 2021: The Pandemic of the Internet*. URL: https://www.imperva.com/resources/resource-library/reports/bad-bot-report/ (visited on 11/12/2021).

[Li+21] Jianhua Li et al. *FLEAM: A Federated Learning Empowered Architecture to Mitigate DDoS in Industrial IoT*. In: *IEEE Transactions on Industrial Informatics* (2021), pp. 1–1. DOI: 10.1109/TII.2021.3088938.

[Naz03] Akif Nazar. *Synthesis & Simulation of Mouse Dynamics*. In: (2003). URL: https://dspace.library.uvic.ca/bitstream/handle/1828/308/Thesis-v19.pdf?sequence=1&isAllowed=y.

[Pap+21] Panagiotis Papadopoulos et al. *ZKSENSE: a Privacy-Preserving Mechanism for Bot Detection in Mobile Devices*. In: *Proceedings on Privacy Enhancing Technologies*. On the Internet: Privacy Enhancing Technologies Symposium, July 2021, pp. 1–23.

[Pyt] Selenium with Python. *Selenium with Python*. URL: https://selenium-python.readthedocs.io/ (visited on 01/25/2022).

[SCG12] Chao Shen, Zhongmin Cai, and Xiaohong Guan. *Continuous authentication for mouse dynamics: A pattern-growth approach*. In: *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2012)*. 2012, pp. 1–12. DOI: 10.1109/DSN.2012.6263955.

[ult] Leon ultrafunkamsterdam. *undetected_chromedriver*. URL: https://github.com/ultrafunkamsterdam/undetected-chromedriver (visited on 02/04/2022).

[Uni] European Union. *General Data Protection Regulation*. URL: %7Bhttps://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32016R0679%7D (visited on 11/12/2021).