Exposé for Bachelor Thesis

How well can federated 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 December 2, 2021

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 explore the state-of-art as well as provide a proof-of-concept for a basic system that incorporates all of the above criterions. These concerns will be addressed by implementing an open-source library that uses federated learning and its performance will be compared against existing solutions.

Contents

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. Especially the increased usage of web interfaces in poorly secured IoT devices and the trend to (re-)implement software as web applications is resposible for this.

The usage of bots can have several goals. 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. The most prominent example are scraper bots from search engines that need to periodically request websites 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 [**TODO**].

Many website operators tend to use solutions that are easy to integrate and perform well. This requires embeding external software which collects user data and sends it to different servers of the software vendors. These often do not disclose 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.

machine learning motivation

A promising technique that combines both machine learning and respecting modern privacy standards is federated learning [KMR15] [Kon+16].

2 Method

we use this and thgis because usw

- Konzept - Datensatz - welche Netze - Evaluierung (wie vergleichen)

This exposé and the thesis based on it initially explore related works and the state of the art in the field of applying machine learning and especially federated learning to bot detection. Next a system architecture of a potential software is proposed as a proof of concept. This includes decisions about what existing software libraries are suitible for integration, how and where the software needs to run in practice, how federated learning will be integrated, and what data exactly is most appropriate to be used in the learning algorithm. A hypothesis is made that this system is practically feasible. The evaluation of the claims is being done in terms of performance, privacy and comparability to existing solutions.

3 | Related Work

3.1 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/

3.2 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 focusses 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.

Other works related to DDoS mitigation that also incorporate machine learning include Farivar *et al.* [Far+20], Liu *et al.* [LLW19] and Hussain *et al.* [Hus+21].

Many of the above solutions use the actual IP network traffic to extract relevant information to be used as input parameters for their models and are intended to be run in a server environment. In comparison clients' browser environments in the web offer a much greater amount and variety of user information that might be very useful to differentiate between a valid user and a malicious bot.

The work of [Pap+21] outlines the problems and privacy-realted concerns really well and tries to solve a very similar problem but focusses 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 proofis generated that is verified on a server.

4 | Proposed Architecture

This work aims to incorporate an existing library that provides federated learning capabilities and can be run either on the client's browser or the websites' servers. Ideally an existing machine learning will be used to limit the scope of the thesis.

The proof of concept is intended to show the feasibility of implementing such a system while taking the specified constraints into account. It will include multiple website instances that all use the bot detection library and a primary server that holds the model, facilitates communication to and from the website instances, and incorporates learned updates.

To avoid having to use user data at all, the learning system could only work on request metadata on the web server or backend level. In this case the available data would be limited to combinations of IP data, such as addresses and timing data, as well as HTTP request metadata which, in most cases, contains pre-defined data that can be easily faked by bots. As [Pap+21] shows, having access to user data from the client's device can be very successful so this work assumes that dynamic user data is needed in order to properly distinguish between human- and bot-initiated requests. It also needs to determine whether it suffices to gather data in the website's backend or frontend only or if both sources are needed. For now all possiblities will remain open and potential existing software will be evaluated.

At the same time a friction-less system is desired that uses already existing data instead of requiring additional user input, e.g. solving a puzzle or performing object detection.

4.1 Website Instance Structure

The thesis will be using the Flower framework because it can use several different machine learning backends for greater flexibility in the choice of network used. It will run as part of the server implementation of the respective websites.

To be determined is whether to perform the actual training of the model in the client's browser or the websites' backend server. The former variant has the advantage of the client data never leaving the own browser which makes a strong case for privacy concerns. It also allows websites that don't use or don't have access to a backend to use this system. Its disadvantages are potentially high performance penalties, e.g. if the training code needs to run on older hardware which also might affect the actual website's performance. Another potential risk is the exposure of the training logic and model which attackers might use to reverse engineer the system and find a way to disguise themselves as non-malicious clients.

Analysis of available libraries revealed only three potential candidates for the browser-based approach:

1. Experimental and unmaintained library for TensorFlow.js [PAI]

2. A github user's attempt [Sah] to solve this problem following a discussion in TensorFlow's github issues.
3. Implementing an own solution on top of a browser-based ML library.

The amount and quality of libraries suitable for the server-side approach seems to be much higher. A selection of the most popular libraries includes the following projects:

1. https://github.com/IBM/federated-learning-lib
2. https://fedml.ai/
3. https://flower.dev/
4. https://www.tensorflow.org/federated/federated_learning

Weighing the advantages and disadvantages of both solutions against each other leads to the choice of running the federated learning client on the website's backends. Both the potentially increased performance and non-exposure of the learning logic and model are the biggest factors in this choice. Privacy-concious website operators need to either choose a hoster that provides access to the backend software, or host the application backend themselves such that control over the user data can be guaranteed. Due to this the requirement to install the software in the backend is reasonable.

4.2 Backend Technology Stack

As python is the de-facto language of choice for machine learning applications and has both great available projects for web applications and is well suitable for rapid prototyping, it is also used for both the website instances' and primary server's backends.

The specific software stack consists of a python application using the Flask microframework which serves a REST-style API and the static frontend files in the website instances.

4.3 Federated Learning

A big part of the system design is going to be the choice of user data and transformations performed on the same to be used in the model's parameter space.

5 Empirical Study

5.1 Performance

To be determined is whether the proposed architecture actually works with the goal.

Bibliography

[Far+20] Faezeh Farivar et al. *Artificial Intelligence for Detection, Estimation, and Compensation of Malicious Attacks in Nonlinear Cyber-Physical Systems and Industrial IoT*. In: *IEEE Transactions on Industrial Informatics* 16.4 (2020), pp. 2716–2725. DOI: 10.1109/TII.2019.2956474.

[Hus+21] Bilal Hussain et al. *Deep Learning-Based DDoS-Attack Detection for Cyber–Physical System Over 5G Network*. In: *IEEE Transactions on Industrial Informatics* 17.2 (2021), pp. 860–870. DOI: 10.1109/TII.2020.2974520.

[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.

[LLW19] Chi Harold Liu, Qiuxia Lin, and Shilin Wen. *Blockchain-Enabled Data Collection and Sharing for Industrial IoT With Deep Reinforcement Learning*. In: *IEEE Transactions on Industrial Informatics* 15.6 (2019), pp. 3516–3526. DOI: 10.1109/TII.2018.2890203.

[PAI] People + AI Research (PAIR). *Federated Learning in TensorFlow.js*. URL: https://github.com/PAIR-code/federated-learning/tree/03deeb2e91c63679bc1ea61cb4af6ede45f69f92 (visited on 10/30/2021).

[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.

[Sah] Shashwat Sahay. *pyjs$_f$ederated$_l$earning*. URL: https://github.com/shashwatsahay/pyjs_federated_learning/tree/31e6f3452909b972fef87d0cdb0d1edaa1e025ed (visited on 10/30/2021).

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