Exposé for Bachelor Thesis

How 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 November 11, 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 which performs well enough to be considered a realistic choid within currently available alternatives.

Contents

1 Motivation

why bot bad

2 Method

This exposé and the thesis based on it will be split into the following high-level parts:

1. Exploration of related work and state of the art
2. Proof of concept of the proposed architecture
3. Empirical study testing the claims
    a) Performance
    b) Privacy
    c) Feasibility

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.

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, fascilitates communication to and from the website instances, and incorporates learned updates.

2.1 Website Instance Structure

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 [**PAIRFL2019**]
2. A github user's attempt [**SaFL2019**] 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.

2.2 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.

3 Related Work

3.1 Proprietary Solutions

https://datadome.co

3.2 Scientific Work

The paper "FLEAM: A Federated Learning Empowered Architecture to Mitigate DDoS in Industrial IoT" [**LiJi2021**] 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.* [**FaFa2020**], Liu *et al.* [**8594641**] and Hussain *et al.* [**9000893**].

Many of the existing state of the art solutions use the actual IP network traffic to extract relevant information to be used as input parameters for their models. The browser environment in the web offers a much greater amount and variety of user information that can be very useful to differentiate between a valid user and a malicious bot.

4 Proposed Architecture

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.

4.1 Federated Learning

asdasd