Botgraph revised

capstone project proposal

team

|  |  |
| --- | --- |
| **Name** | **Registration Number** |
| Mohsin Ashraf | 15140094 |
| Mushood Hanif | 14140062 |
| Mehwish Hameed | 15140107 |
| Rida Yaqoob | 14140044 |

Table of Contents

[**Jargon** 3](#_Toc4592601)

[**Problem Statement** 5](#_Toc4592602)

[**Introduction** 5](#_Toc4592603)

[**Solution** 5](#_Toc4592604)

[**Introduction** 5](#_Toc4592605)

[**Procedure** 6](#_Toc4592606)

[**Scope** 6](#_Toc4592607)

[**References** 7](#_Toc4592608)

# **Jargon**

|  |  |
| --- | --- |
| **Word** | **Meaning** |
| DDoS | Distributed Denial-of-Service attacks where multiple systems are infected with a virus to target a single system causing Denial of Service to legitimate users. |
| Bots | An autonomous program on a network which can interact with systems or users, designed to behave like a user. |
| Signature-based solutions | An approach that identifies the presence of a bot by identifying, comparing and matching at least the code pattern of the software in question with the database of signatures of known bots. |
| IP-based solutions | A rate-limiting algorithm is used to check whether an IP-address has to be placed in checks based on the information in the session cache. In case an IP makes too many requests within a given timeframe, the server will respond with and error saying “Too Many Requests” |
| Headless Browser | A browser without a GUI, i.e. a browser that has internet access but doesn’t display them on screen. |
| Authentication | The process of verifying the identity of a user or process. |
| Social Bots | Bots that specialize only in targeting social media and networks. |
| Web Bots | Bots that specialize in targeting networks that do not possess the concept of user account. |
| Sitemap | It is a directed graph in which the nodes represent a URL pattern e.g.: /page?id=\* and each edge represents the hyperlink that links 2 URLs together. |
| Active Crawling | Active crawling requires to run a crawler to build the sitemap of the site. The crawling typically starts from the homepage and enters each hyperlink from the current page recursively. Each URL pattern is retrieved only once to reduce the number of pages need crawling. This is based on the assumption that web pages with same URL pattern have the same page structure and similar hyperlinks of the same URL patterns. |
| Hyperlinks | It is a reference to data that a user can directly follow. |
| HTML Request Behaviour Fields | A request is basically used to represent the HTTP request sent to the server from the client. A request has many defining properties that can be classified into 2: identity fields which are used to identify either the client or the server and behavior fields that define the access behavior of the client. The following are the request fields:   |  |  | | --- | --- | | Timestamp | Request time, e.g., 2019-1-12 04:00:07 | | HttpMethod | HTTP request method, e.g., GET, POST. | | **RequestUri** | **The path in URL, e.g., /books/desc?id=1** | | **Status** | **HTTP status code, e.g., 200, 404.** | | Host | “Host” field in request header. | | UserAgent | “User-Agent” field in request header. | | ClientIp | Client’s IP address. | | Timestamp | Request time, e.g., 2019-1-12 04:00:07 |   The fields in bold format are fields that determine the client’s behavior. |
| Status Code | The request has a behavior property called *Status* that returns a value which depicts the access level of the client and also the availability of the data that the URL leads to. |
| Model | A framework used to detect whether the client is a bot or not. |
| CNN | **C**onvolutional **N**eural **N**etworks. |
| Topology | In the context of neural networks, topology refers to the way the neurons are connected in a neural network. |
| Evolutionary Techniques | Techniques that mimic or implement Darwinian evolution. |

# **Problem Statement**

## **Introduction**

In 2016, Incapsula, A cloud-based application delivery platform generated a report regarding its internet traffic and found out that 51.8% of its traffic came from automated web bots including search engines, price scrappers, and viruses which cause **DDoS** attacks.These bots not only cause the leakage of business data, but also consume significant bandwidth and server overload.

**Bots** have been blamed for consuming a lot of internet traffic for some time now. Traditional bot detection studies focus on **signature-based** solutions but bots now forge identities to bypass detection through this method. There are **IP-based** solutions where each time a bot is detected the IP it corresponds to, gets blacklisted but that introduces an unnecessary element of maintaining a database of blacklisted IPs. Also, to bypass this, proxies and VPNs are easily available to mask and use an IP Address. Detection things like mouse movement and using JavaScript validation engines can be used but bots can now bypass all of this with the help of environments like **Headless Chrome.**

# **Solution**

## **Introduction**

Based on whether **authentication** is required or not, bots can be classified broadly into 2 categories: **social bots,** which target social networks and **web bots,** which target general websites. Compared to social bots, web bots are harder to detect because unlike social networks, there is no concept like the user account in the web traffic, and for a cloud provider, there must be millions of websites providing distinct services to its customers. Therefore, to efficiently identify bots among the websites, a generalized approach must be formulated for heterogenous scenarios.

## **Procedure**

BotGraph basically is performed in the following steps:

* Firstly, a **sitemap** is defined.
  + We will use **Active Crawling** to create a crawler that recursively visits all the possible **Hyperlinks** in all the pages of the website to finalize the sitemap of the website targeted.
* Each user session is mapped to a subgraph of the sitemap which contains all the URLs the user has visited and their corresponding frequency of visiting.
  + We will get specific **HTML Request Behaviour Fields** from the server namely: *Request Uri, Status* where the *Request Uri* will be used to highlight nodes in the sitemap according to user activity on the website and the *Status* will help determine whether the user request placed is valid or not. Validity of a request is decided on the basis of **Status Codes** returned by the *Status* Field and some of its values correspond to errors which would be considered as invalid. All the invalid requests would be mapped onto a single node named INVALID to keep track of how the website responds to certain requests for a user.
* 2D images of the subgraphs based on the sitemap are generated to create a dataset on which our **model** will then train.
* A state-of-the-art **CNN** would be written from scratch and its **topology** would be optimized using **Evolutionary Techniques** to surpass the performance of existing CNNs.
  + \*optimization techniques
* The newly designed CNN would be trained on the subgraph dataset to classify and identify whether the subgraph is “bot” or “non-bot”.

# **Scope**

We would be concentrating on identifying web bots on e-commerce websites for demonstration purposes but our approach is generalized and can work on any non-social network. As mentioned above, we would be creating our own dataset for training our custom written CNN. We would also be writing crawlers, one for active crawling and the other for subgraph plotting. We might also write web bots for testing purposes. The algorithm that we would be using to detect web bots would be a self-designed CNN whose topology would be optimized using evolutionary techniques.

# **References**

**BotGraph: Web Bot Detection Based on Sitemap  
Yang Luo**, **Guozhen She**, **Jinwan Huang**, **Peng Cheng** and **Yongqiang Xiong**  
Microsoft Research  
Fudan University  
Beihang University