05/01/2023

Web surfing bots against Bot detectors

Jay Patel (jsp202)
Jay Kania (jpk164)

Abstract:

The objective of this project was to develop a unique algorithm capable of distinguishing between human-generated and robot-generated website traffic. Unlike other approaches, this algorithm focuses on analyzing time intervals between requests as the primary basis for classification. Human-generated traffic was simulated using heuristic data. The project aimed to assist web administrators in identifying and managing unwanted traffic on their networks or resources, while also providing a tool to estimate the proportion of human traffic versus robot traffic. By relying solely on server access logs rather than captured network packets, the algorithm offers a realistic representation of the data accessible to web administrators for analysis. This project has various practical applications, including determining the percentage of traffic that contributes to ad revenue. Additionally, it provided the opportunity for the team to explore web scraper construction, understand its ease of use, and uncover the challenges associated with traffic classification.

Introduction:

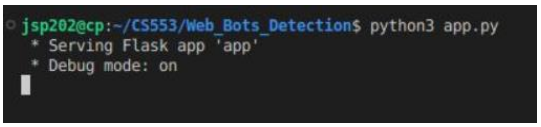
Our project aimed to develop a unique algorithm capable of distinguishing between human-generated and robot-generated HTTP request logs from a web server. The target audience for our tools and findings included Web Admins, Network Admins, Commercial, and educational stakeholders. To accomplish this, we designed and implemented a custom web server hosting a website comprising 10 interconnected web pages. Each HTTP request made during a session was logged, along with a timestamp, creating a log file that served as the input for our detection algorithm. This algorithm predicted whether the logged traffic was generated by a human or a robot. To generate various types of traffic, we created progressively sophisticated web scrapers. For each iteration of the web scraper, we modified the detection algorithm to enhance its ability to differentiate human traffic from robot traffic. The algorithm primarily relied on analyzing the properties of time intervals between requests to make predictions. To evaluate the effectiveness of our system, we labeled the human-generated and robot-generated traffic logs separately, enabling the detection algorithm to track the number of correct predictions. We tested each version of the detection algorithm against traffic logs generated by different versions of the web scrapers, as well as against human traffic. By adopting this approach, we aimed to provide a novel solution for detecting and categorizing web traffic, benefiting a wide range of professionals and researchers in various domains.

Methodology:

To create the web server and start logging traffic, we leveraged a Python program and utilized the Flask library. The implementation involved incorporating system logging to direct the logs to a

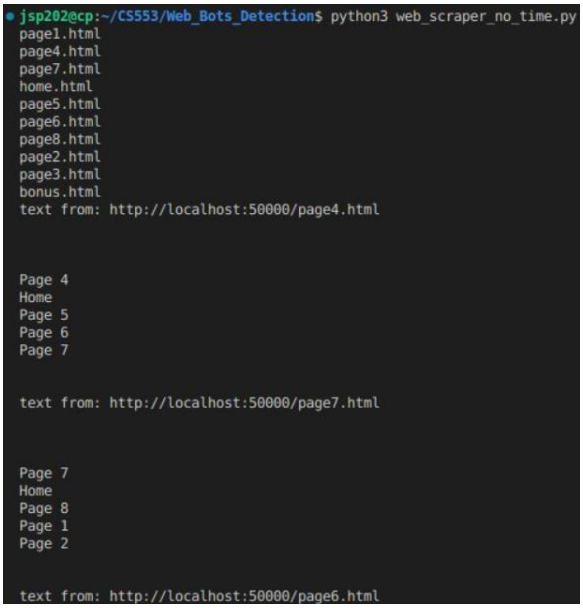
file instead of the console. Running the server is a straightforward process where users execute the Python script. Below, you can find the command and its corresponding output:

Python3 app.py



To generate traffic for our project, there are two options available to users. They can either visit the website directly by accessing http://localhost:50000 and interact with the pages by clicking on various links, or they can utilize one of the web scraper scripts provided. To simulate human traffic, we manually browsed the website, attempting to replicate typical user behavior observed on various types of websites such as blogs, documentation, and e-commerce sites. Each session was intentionally designed to have variations in the number of pages visited, the order of pages, and the duration spent on each page.

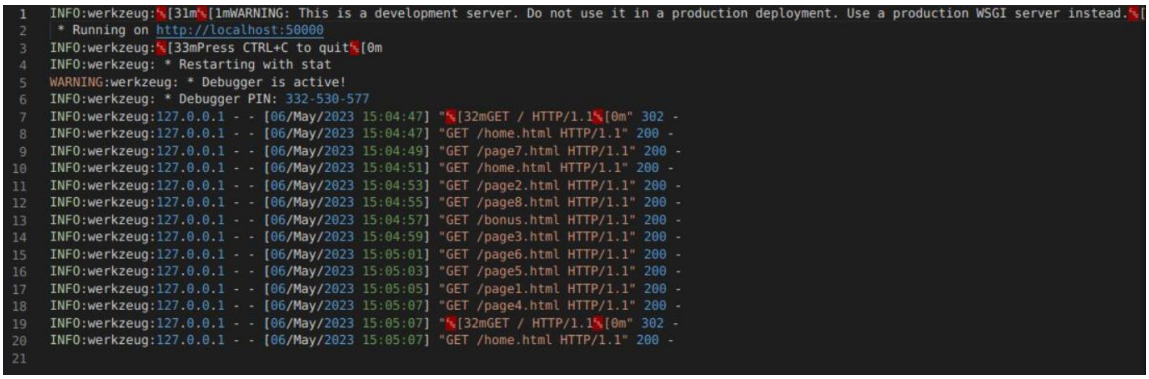
To construct the web scraper, we utilized the "requests" library and the Beautiful Soup library in Python. The "requests" library facilitated requesting web pages, while Beautiful Soup enabled the parsing of HTML files. The initial version of our web scraper was designed to request the main page, parse its HTML document for embedded links, visit each of those pages randomly, parse them for additional links, and so on, until all links were visited. The scraper would then display the link and the content of the page on the console. Here's an example output from one of the scraper scripts:



In the subsequent iteration of the web scraper, we maintained the same behavior of visiting pages and parsing their content, but introduced a deliberate two-second delay between each request. This delay aimed to simulate a more realistic browsing pattern, where humans typically take some time between page interactions.

In the final iteration of the web scraper, we further enhanced its behavior by incorporating a randomized delay between requests. Prior to each request, a random number was generated, ranging from 0 to 3 seconds. This variation in delay time aimed to mimic the unpredictable nature of human browsing habits.

To capture the generated traffic and create log files for analysis, we followed a systematic process. First, we started the server to enable logging. Then, we ran the web scraper or generated human traffic by interacting with the website. Once the desired traffic generation session was complete, we closed the server and labeled the corresponding log file according to how the traffic was generated. For each session, a separate log file was recorded, ensuring clear distinction and traceability. Here's an example of the generated log files:



Each log file contained the recorded HTTP requests, timestamps, and any additional relevant information that could aid in subsequent analysis and labeling for the detection algorithm.

By capturing and categorizing the traffic in this manner, we aimed to establish a robust dataset for evaluating and refining the detection algorithm's performance.

The generated logs served as input for the detection algorithm, which underwent several iterations to classify the traffic as either human or robot. Here's an overview of each version of the detection algorithm and its classification approach:

1. Initial Detection Algorithm (against a Basic Scraper):
    a. Metric: Number of unique delays between requests
    b. Classification Criteria: If the number of unique delays was greater than threshold (2), the traffic was classified as human.
2. Enhanced Detection Algorithm (against a Constant Delayed Scraper):
    a. Metric: Mode of request intervals and the ratio of mode values to total intervals
    b. Classification Criteria: If the ratio of mode values to total intervals exceeded threshold (0.75), the traffic was classified as a robot.
3. Final Detection Algorithm (against a Randomly Delayed Scraper):
    a. Metric: Standard deviation of request intervals
    b. Classification Criteria: If abs(1 - standard deviation) was less than or equal to threshold (0.2), the traffic was classified as a robot.

These thresholds and classification criteria were established based on observed values from human and robot data. The detection algorithm accepted a folder of log files as input, classifying each file accordingly. It also kept track of the correctness of each classification and provided a final success rate which indicated the accuracy of our algorithm.

The output displayed the classification results for each log file, indicating whether the classification was correct or incorrect. Finally, the algorithm provided an overall accuracy percentage reflecting the correctness of its classifications.

This iterative process of the detection algorithm allowed for refinement and improvement in accurately classifying traffic as human or robot, contributing to the overall effectiveness of the system.

Testing:

We fed the detection algorithm with an equal number of human-generated and bot-generated log files. The detection algorithm checked whether the file met the human threshold or not. If the threshold is met, the traffic is classified, i.e., labeled as bot-generated; else, labeled as human-generated.

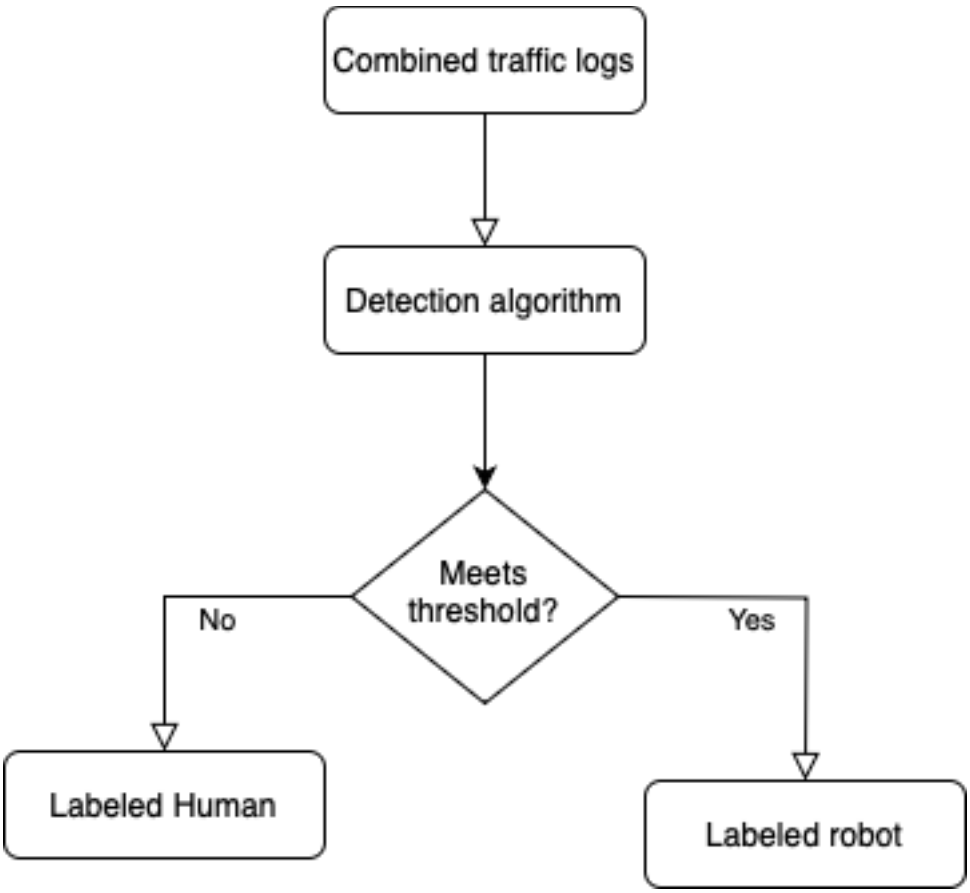
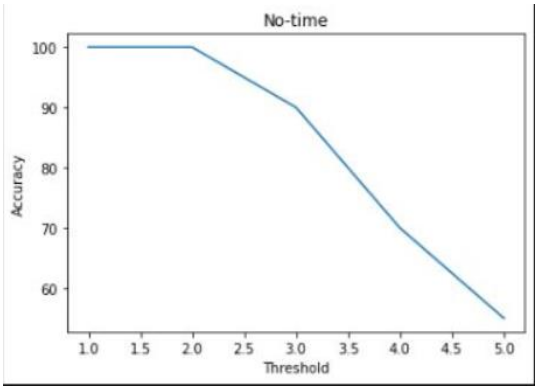


Figure: Flow diagram for testing

1. No time result:

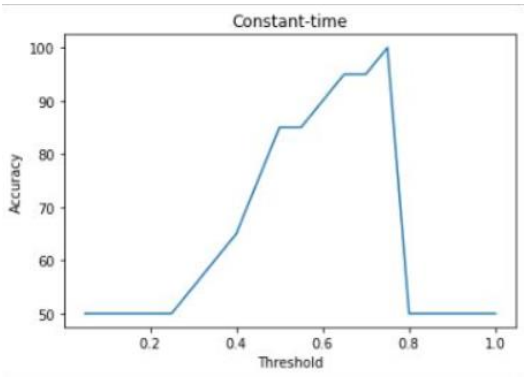
Threshold	Accuracy (%)
1	100
2	100
3	90
4	70
5	55




```
Incorrectly identified: human_log_4.log
Incorrectly identified: human_log_3.log
Correctly identified: bot_no_time_log_7.log
Correctly identified: bot_no_time_log_9.log
Incorrectly identified: human_log_2.log
Correctly identified: human_log_5.log
Correctly identified: bot_no_time_log_8.log
Correctly identified: bot_no_time_log_6.log
Correctly identified: bot_no_time_log_1.log
Correctly identified: bot_no_time_log_2.log
Correctly identified: bot_no_time_log_5.log
Incorrectly identified: human_log_10.log
Incorrectly identified: human_log_6.log
Incorrectly identified: human_log_1.log
Incorrectly identified: human_log_8.log
Correctly identified: bot_no_time_log_4.log
Correctly identified: bot_no_time_log_3.log
Correctly identified: bot_no_time_log_10.log
Incorrectly identified: human_log_9.log
Incorrectly identified: human_log_7.log
success rate: 55.00000000000001%
```

2. Constant time result:

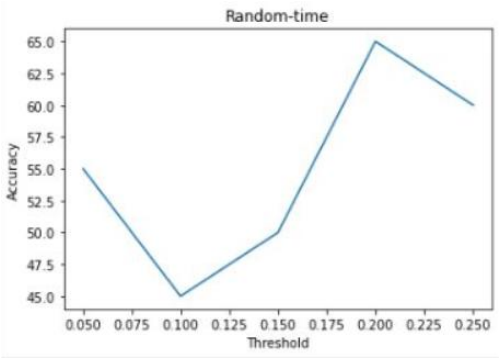
Threshold	Accuracy (%)
0.05	50
0.10	50
0.15	50
0.20	50
0.25	50
0.30	55
0.35	60
0.40	65
0.45	75
0.50	85
0.55	85
0.60	90
0.65	95
0.70	95
0.75	100
0.80	50
0.85	50
0.90	50
0.95	50
1.00	50




```
Correctly identified: human_log_4.log
Correctly identified: human_log_3.log
Correctly identified: bot_constant_time_log_3.log
Correctly identified: bot_constant_time_log_4.log
Correctly identified: human_log_2.log
Correctly identified: human_log_5.log
Correctly identified: bot_constant_time_log_5.log
Correctly identified: bot_constant_time_log_2.log
Correctly identified: bot_constant_time_log_1.log
Correctly identified: bot_constant_time_log_6.log
Correctly identified: bot_constant_time_log_8.log
Correctly identified: human_log_10.log
Correctly identified: human_log_6.log
Correctly identified: human_log_1.log
Correctly identified: human_log_8.log
Correctly identified: bot_constant_time_log_9.log
Correctly identified: bot_constant_time_log_7.log
Correctly identified: human_log_9.log
Correctly identified: human_log_7.log
Correctly identified: bot_constant_time_log_10.log
success rate: 100.0%
```

3. Random time result:

Threshold	Accuracy (%)
0.05	55
0.10	45
0.15	50
0.20	65
0.25	60




```
jsp202@cp:~/CS553/Web_Bots_Detection$ python3 bot_detector_random_time.py
Correctly identified: human_log_4.log
Incorrectly identified: human_log_3.log
Correctly identified: bot_random_time_log_3.log
Correctly identified: bot_random_time_log_4.log
Correctly identified: human_log_2.log
Correctly identified: human_log_5.log
Correctly identified: bot_random_time_log_5.log
Correctly identified: bot_random_time_log_2.log
Correctly identified: bot_random_time_log_8.log
Correctly identified: bot_random_time_log_1.log
Correctly identified: bot_random_time_log_6.log
Incorrectly identified: human_log_10.log
Incorrectly identified: human_log_6.log
Incorrectly identified: human_log_1.log
Correctly identified: human_log_8.log
Correctly identified: bot_random_time_log_7.log
Correctly identified: bot_random_time_log_9.log
Incorrectly identified: bot_random_time_log_10.log
Incorrectly identified: human_log_9.log
Incorrectly identified: human_log_7.log
success rate: 65.0%
```

Results:

During the evaluation of the detection algorithm, it was observed that the algorithm performed well in detecting simpler versions of bots, such as the basic scraper and statically delayed bots. These bots exhibited consistent patterns that made them distinguishable from human behavior, allowing the algorithm to classify them accurately.

However, the algorithm encountered challenges in accurately classifying human versus bot behavior when dealing with the randomly delayed bot. The increased variability introduced by the random delays made it more difficult for the algorithm to discern between human-like browsing patterns and the bot's behavior. As a result, the detection algorithm may have struggled to achieve the same level of accuracy in distinguishing between human and bot traffic in this specific case.

This observation aligns with the nature of evolving bot technology, where newer generations of bots often aim to overcome existing detection measures, making it difficult for older detection models to maintain high accuracy rates.

It suggests that further refinement of the detection algorithm may be necessary to improve its performance in accurately classifying such traffic scenarios. Adjusting the classification criteria or incorporating additional metrics could potentially enhance the algorithm's ability to differentiate between human and randomly delayed bot traffic.

It is worth noting that the "advanced detection algorithm" achieved correct detection of 90% of all robot traffic, demonstrating its effectiveness in identifying the latest bot behaviors. However, it came with a trade-off, as it inaccurately classified some of human traffic as bots. This highlights the potential for false positives, where legitimate human traffic is mistakenly labeled as bot traffic, which can have significant implications, such as blocking genuine users.

This difficulty in accurately classifying human traffic and minimizing false positives underscores the complexity of the task and the need for ongoing improvement and fine-tuning of detection algorithms. Striking the right balance between accurately identifying bot traffic and avoiding false positives is a persistent challenge in the field of traffic classification and highlights the importance of continuous research and development in this area.

Conclusions:

In conclusion, the results and process of building this system have provided valuable insights into the challenges and complexities of detecting and classifying bot behavior. One key takeaway is the relative ease with which detection measures can be evaded, highlighting the ongoing "arms race" between detection and evasion techniques. Accurately classifying bot behavior while minimizing false positives remains a challenging task.

Furthermore, this project has underscored the simplicity of constructing web scrapers to gather information from websites. The availability of tools and libraries facilitates the development of advanced scraping functionality, with the main focus being on tailoring the scraper to the target website and the specific information sought.

In order to enhance future iterations of this project, several improvements can be considered. First, incorporating real-world human logs captured from actual browsing habits would provide more representative examples and insights into human traffic patterns. While self-generated logs were used in this project, leveraging concrete examples would offer more instructive data.

Additionally, expanding the sample size of both human and robot logs would contribute to a more comprehensive analysis and evaluation of the detection algorithms. This would provide a broader understanding of the system's performance across a range of scenarios.

Furthermore, utilizing a web server capable of generating more granular logs, and capturing timestamps at the millisecond or microsecond level, would be beneficial. This would enable the

detection algorithms to analyze finer access patterns and provide users with more precise tools for evaluation.

By addressing these areas of improvement, future iterations of this project could enhance the accuracy and robustness of the detection algorithms, further contributing to the field of traffic classification and assisting web administrators in effectively identifying and managing unwanted traffic on their networks.

References:

[1] "Welcome to flask," *Welcome to Flask - Flask Documentation (2.2.x)*. [Online]. Available: https://flask.palletsprojects.com/en/2.2.x/. [Accessed: 07-May-2023].

[2] "Beautiful Soup documentation," *Beautiful Soup Documentation - Beautiful Soup 4.12.0 documentation*. [Online]. Available: https://www.crummy.com/software/BeautifulSoup/bs4/doc/. [Accessed: 07-May-2023].

[3] H. Xu, Z. Li, C. Chu, Y. Chen, Y. Yang, H. Lu, H. Wang, and A. Stavrou, "Detecting and characterizing web bot traffic in a large e-commerce marketplace," *Computer Security*, pp. 143–163, 2018.

[4] G. Suchacka, A. Cabri, S. Rovetta, and F. Masulli, "Efficient on-the-fly web bot detection," *Knowledge-Based Systems*, vol. 223, p. 107074, 2021.

Appendix:

1. GitHub link: https://github.com/Jayp13997/Web_Bots_Detection

Steps to implement the project:

1. Please download the entire source code from the GitHub link provided above.
2. Keep all the files in the same directory.
3. Run these files in the following order:
    a. To set up the server:
        i. app.py
    b. Log generation:
        i. For web scraper generated logs: web_scraper_no_time.py for no time delays, web_scraper_constant_time.py for constant time delay and web_scraper_random_time.py for random time delay between HTTP requests (make sure to change the file name in app.py).
        ii. For manual/human logs: http://localhost:8000
    c. To classify the generated logs:
        i. bot_detector_no_time.py, bot_detector_constant_time.py, and bot_detector_random_time.py for each time delay type.

Failure parts:

We believe in highlighting both our achievements and our setbacks. Throughout our project, we encountered a couple of significant challenges:

1. Attempting to host the website for external access:

   Initially, we tried hosting everything on the ilab as a personal web network. Unfortunately, our attempts were met with naivety and limited success because of us not being able to have access to get logging working. You can visit people.cs.rutgers.edu/~jpk164 to see our early endeavors toward the project.

2. Exploring a DOM-based network hosted on free-of-cost 3rd party servers:

   While investigating the potential of a DOM-based network and utilizing free hosting services, we faced a persistent issue. Our process was repeatedly terminated, resulting in the cancellation of all on-click event logs.

It's important to acknowledge that these challenges provided valuable insights into the complexities of web hosting and the limitations of certain approaches. Rather than being discouraged, we view these experiences as learning opportunities that contribute to our growth and understanding.

Moving forward, we recognize the importance of carefully evaluating hosting options, understanding their limitations, and seeking alternative solutions when necessary. By leveraging these lessons, we aim to enhance our future endeavors and ensure the successful execution of our projects.