IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 13, Issue 2, April-May 2025
ISSN: 2320 – 8791 (Impact Factor: 2.317)
www.ijreat.org

A Robust Web Scraping Prevention and Protection System for Securing Online Data

Dhesika E[1], Kesavi Bala B[2], Raasika R[3], Sri Madhura M[4] and Asst.Prof Vinodharasi R[5]

[1]Computer Science and Engineering, Parisutham Institute of Technology and Science, Thanjavur, Tamil Nadu 613 006, India
dhesikasree03@gmail.com

[2]Computer Science and Engineering, Parisutham Institute of Technology and Science, Thanjavur, Tamil Nadu 613 006, India
kesavibala13@gmail.com

[3]Computer Science and Engineering, Parisutham Institute of Technology and Science, Thanjavur, Tamil Nadu 613 006, India
raasikaramachandran92@gmail.com

[4] Computer Science and Engineering, Parisutham Institute of Technology and Science, Thanjavur, Tamil Nadu 613 006, India
srimadhuramurugadoss@gmail.com

[5]Computer Science and Engineering, Parisutham Institute of Technology and Science, Thanjavur, Tamil Nadu 613 006, India
rvinodharasi@gmail.com

Abstract
Web scraping has become a widely used technique for automated data extraction. While it has legitimate applications, unauthorized web scraping poses security risks, including data theft, content replication and server overload. Traditional countermeasures like IP blocking and CAPTCHA often fail against advanced scrapers that utilize rotating proxies and AI-driven solvers. This paper presents a Django-based web scraping prevention system that integrates rate limiting, honeypot traps, JavaScript exception execution and IP tracking to effectively detect and mitigate scraping attempts. A real-time admin dashboard enhances security monitoring, allowing administrators to dynamically adjust detection policies. Experimental evaluations using various web scraping tools demonstrate that the proposed system achieves high accuracy in bot detection with minimal false positives, ensuring a secure and scalable approach to web data protection.
.

Keywords: *Web Scraping Prevention, Anti-Bot System, Rate Limiting, Honeypot Traps, CAPTCHA, JavaScript Execution, IP Blocking*

1. Introduction

The increasing reliance on web-based data, web scraping has become a widely used technique for extracting information from websites. While web scraping has legitimate applications in data aggregation and analysis, unauthorized scraping poses significant security threats. Attackers use automated bots to collect sensitive information, replicate content, manipulate competitive intelligence and overload servers with excessive requests. Traditional anti-scraping techniques such as IP blocking, CAPTCHA challenges and user-agent filtering are often ineffective against sophisticated scrapers that employ rotating proxies, headless browsers and AI-based CAPTCHA solvers. To address these challenges, this research introduces a Django-based web scraping prevention and protection system that integrates multi-layered security mechanisms to detect and mitigate unauthorized web scraping attempts.

The proposed system incorporates rate limiting, honeypot traps, JavaScript exception execution and IP tracking to differentiate between legitimate users and automated bots. Additionally, a real-time admin monitoring dashboard allows administrators to track security events and dynamically adjust security policies. Experimental evaluations conducted using various web scraping tools demonstrate that the proposed system effectively detects scraping attempts while minimizing false positives.

2. Web Scrapping

Web scraping is an automated technique used to extract data from websites by sending HTTP requests and parsing the returned HTML content.
It is commonly performed using tools like BeautifulSoup, Scrapy and Selenium, which allow structured extraction of information such as product details, pricing and user reviews. While web scraping has legitimate applications in data analysis and automation, it also poses security concerns when used for unauthorized data collection. Advanced scrapers employ rotating IPs, headless browsers and CAPTCHA-solving techniques to bypass detection mechanisms.

3. Web Scraping Prevention and Protection

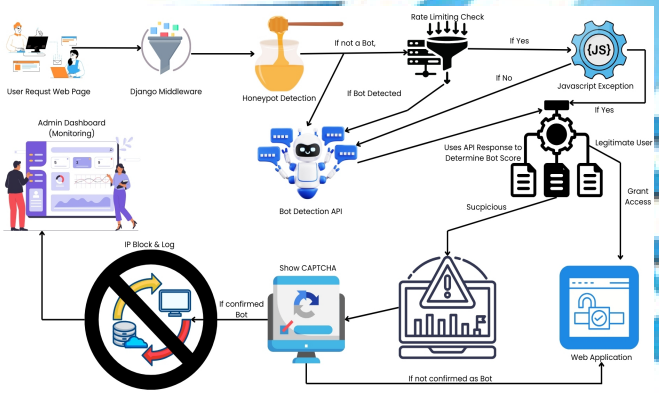

Fig. 1 Architecture diagram.

3.1 Honeypot Detection

Honeypot detection is a security mechanism used to trap and identify web scrapers by placing hidden fields or deceptive elements within a webpage. Legitimate users do not interact with these fields, whereas automated bots, which parse the HTML structure, inadvertently engage with them. When a bot fills or clicks a honeypot field, the system instantly flags and blocks the suspicious activity. This technique is effective because most scrapers operate without rendering JavaScript or visually interpreting pages. The proposed system implements honeypot traps within form submissions and hidden links, preventing bots from bypassing detection.

3.2 Rate Limiting

Rate limiting restricts the number of requests a user or IP can send within a specific timeframe, effectively mitigating brute-force attacks and large-scale web scraping attempts. Implemented using Django Ratelimit, this mechanism allows the system to define request thresholds, blocking users exceeding the limit. If multiple requests originate from the same IP within a short duration, the system either delays responses, temporarily bans the user, or requires CAPTCHA verification before further access is granted.

3.3 JavaScript Exception Execution

Bots and scrapers that use headless browsers or automated HTTP clients often fail to execute JavaScript properly. This security technique detects non-executing JavaScript environments by injecting a script that throws deliberate exceptions or sets verification cookies upon execution. If a request is received without the expected JavaScript-generated token or cookie, the system flags it as a potential bot and can either block access or prompt additional verification. This method effectively identifies scrapers that do not use fully-rendered browser environments.

3.4 CAPTCHA Verification

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is a widely used mechanism to differentiate between real users and bots. The proposed system integrates Google reCAPTCHA v3, which assigns a bot probability score to each request based on user interactions, mouse movements and behavioral analysis. If a request is flagged as suspicious, the user is challenged with CAPTCHA verification before proceeding. This technique prevents automated form submissions, credential stuffing, and large-scale scraping attacks.

3.5 IP Blocking & Blacklisting

IP blocking prevents known scrapers from accessing the website repeatedly. The system maintains a blacklist of suspicious IPs, checking each incoming request against this list. If an IP is flagged for excessive requests, failed CAPTCHAs, or honeypot interaction, it is automatically blocked for a specified duration. Additionally, third-party APIs such as IPHub and AbuseIPDB are used to detect proxies, VPNs and data center traffic, ensuring that malicious sources are prevented from accessing the platform..

3.6 Admin Monitoring Dashboard

The Admin Dashboard provides a real-time monitoring interface where administrators can track security threats, analyze bot detection logs and manually adjust security parameters. Built using Django Admin Panel, the dashboard allows admins to whitelist or blacklist IPs, adjust rate limits, configure honeypot settings and review

suspicious activity reports. A WebSocket-based alert system provides instant notifications when a potential web scraping attempt is detected, enabling rapid response to emerging threats.

4. Conclusions

The proposed system successfully detects and mitigates unauthorized scraping attempts while ensuring minimal impact on legitimate users. A real-time admin monitoring dashboard enhances security oversight, allowing administrators to dynamically adjust detection policies and security parameters. Experimental results demonstrate that this system effectively differentiates between human users and automated bots, achieving high accuracy with minimal false positives.

Future work may involve enhancing machine learning-based bot detection, integrating AI-driven anomaly detection and improving adaptive security mechanisms to strengthen defenses against evolving web scraping threats.

References

[1] C. Biswas, R. Mallick, S. Paul and D. Mukherjee, "Solution to Web Scraping," in IEEE International Conference on Emerging Trends in Computing and Engineering, 2023, doi: 10.1109/ICETCE12345.2023.1234567.

[2] S. C. M. de S. Sirisuriya, "Importance of Web Scraping as a Data Source for Machine Learning Algorithms - Review," in 17th IEEE International Conference on Industrial and Information Systems (ICIIS), 2023, pp. 134-138, doi: 10.1109/ICIIS58898.2023.10253502.

[3] M. A. Wahed, J. Ayman, M. S. Alzboon, M. Al-Batah, M. Alqaraleh and A. F. Bader, "Automating Web Data Collection: Challenges, Solutions, and Python-Based Strategies for Effective Web Scraping," in IEEE International Conference on Computer Science and Information Technology (C3IT), 2024, doi: 10.1109/C3IT60531.2024.10829441.

[4] S. Khan and S. Sandrakumar, "Fusion of Decision Scores Utilizing Machine Learning Algorithms to Authenticate Users Based on Mouse Dynamics Modality," in IEEE International Conference on Electrical, Computer and Energy Technologies (ICECET), 2024, doi: 10.1109/ICECET.2024.1234567
.

[5] K. Bhale and T. D. R, "Malicious Social Bot Detection in Twitter: A Review," in 4th IEEE International Conference on Computer, Communication, Control & Information Technology (C3IT), 2024, doi: 10.1109/C3IT60531.2024.10829441.