Web Scraping

https://campus.barracuda.com/doc/98225910/

Web scraping involves copying large amounts of data from a website or application using automated tools, often for commercial advantages that are to the detriment of the organization that owns the web application. Typically, the motivation of the attacker is to undercut the competition, steal leads, hijack marketing campaigns, and appropriate data via the web application. Examples include theft of intellectual property from digital publishers, scraping products and pricing information from e-commerce sites, and stealing listings on real estate, auto dealers, and travel sites.

There are a variety of automated tools, products, and services available for web scraping that can extract data and metadata from web applications as well as from web-based APIs. Advanced tools can even automatically navigate to pages behind forms by automatically filling them in.

Their navigation and extraction features make scrapers very similar to search engines that also intend to index the whole site. Unlike search engines that drive prospects to businesses, scrapers intend to take away business from the sites they are scraping. This makes it important for a security solution to be able to distinguish between genuine search engines and web scrapers, even when some scrapers fake their identity as search engines.

To prevent your web applications from being scraped, configure the web scraping policy on the Barracuda WAF-as-a-Service.

Configuring Web Scraping Policy

The web scraping policy provides the following settings:

Insert Hidden Links in Response
When enabled, the Barracuda WAF-as-a-Service embeds a hidden link in the response. The embedded link does not get displayed on the browser, so a human browsing the web pages through a common browser should never see and click the hidden link. Therefore, any request that attempts to access the hidden link is identified as an automated bot or scraper.

Insert Disallowed URLs in Robots.txt
Typically, every website includes a “/robots.txt” file that provides access instructions such as the user agents that are allowed to access the site, and the web pages that are allowed/disallowed to be accessed by bots.

Example:
User-agent: *

Disallow: /researchtools/abc/

Here, User-agent : Asterisk (*) is a wildcard character and indicates that this website can be accessed by all bots, and Disallow : /researchtools/abc/ indicates that the bots are not allowed to access the /researchtools/abc/ page on the website.

When **Insert Disallowed URLs in Robots.txt** is set to **Yes**, the Barracuda WAF-as-a-Service inserts an encrypted URL into the robots.txt file under Disallow. Any bot that tries to access the encrypted URL is identified as a bad bot and is added to the Block List.

Insert JavaScript in Response
When enabled, the Barracuda WAF-as-a-Service inserts a JavaScript in the response. If the request is from a client (web browser), the JavaScript gets executed and returns with a value for the cookie. If the JavaScript fails to execute, then the client is marked as a bot and added to the Block List.

Insert Delay in Robots.txt
You can slow down the requests from a bot to a web application by setting the delay time (in seconds) between subsequent requests so that server resources are not consumed and are accessible for legitimate traffic.

When **Insert Delay in Robots.txt** is set to **Yes**, the Barracuda WAF-as-a-Service automatically inserts “crawl-delay” in the robots.txt file with the specified Delay Time. All good bots should honor the delay time specified in the robots.txt file while accessing the web application. If not, it is identified as a bad bot and is added to the Block List.

Allowed Bots
Create a list of search engine bots that you want to allow access to your web application by providing the User Agent and Host value pair. For example: User Agent: googlebot and Host: *.google.com.

When a client identifies itself as a search engine via the User Agent field, the system performs a reverse DNS lookup (rDNS) on the source IP address, which yields the true domain associated with the IP address. If this domain does not match the Host value configured above, then the client is classified as a fake bot and web scraping policies are applied on the request. If the configured Host value matches the rDNS domain value, then the request is exempted from further web scraping validation.

Steps to Configure a Web Scraping Policy

1. On the WAF-as-a-Service web interface, go to the **APPLICATIONS** page and click on the application to which you want to configure the Web Scraping policy.
2. On your application page, click **DDoS** in the left panel and then click **Web Scraping**.
3. On the **Web Scraping** page, do the following:
    1. **Web Scraping Protection** – Set to **ON** to enable Web Scraping policy.
    2. **Insert hidden links in response** – Set to **ON** to enable Barracuda WAF-as-a-Service to insert hidden links in the response.

3. **Insert disallowed URLs in robots.txt** – Set to **ON** to enable Barracuda WAF-as-a-Service to insert encrypted URL into the robots.txt file under Disallow.
4. **Insert JavaScript in response** – Set to **ON** to enable Barracuda WAF-as-a-Service to insert JavaScript in the response.
5. **Insert delay in robots.txt** – Set to **ON** to enable Barracuda WAF-as-a-Service to insert “crawl-delay” in the robots.txt file with the specified Delay Time.
6. Click **Save**.