

REPORT FOR RESEARCH IN THE ANTIBOT SYSTEM OF DELTA

Theodoros Bogiatzis

09/23/2025

Domain: Delta.com

Antibot Software: Akamai

Objective: Identify the offers API, obtain a valid JSON response and reverse engineer antibot defenses

Executive Summary

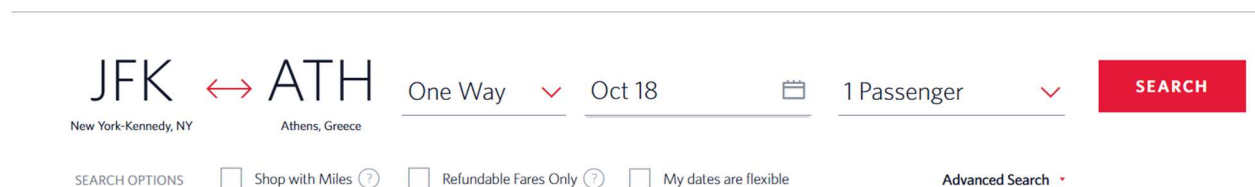
This research analyzed and bypassed Delta's antibot protections (Powered by Akamai) to access the flight orders API. By capturing browser traffic, replicating request using curl (Linux) and httpClientRequest, Invoke-http Request (PowerShell, Windows 11), and systematically testing different headers and client behaviors, I determined that the antibot enforces specific headers and rejects chunked transfer encoding, but does not validate cookies, TLS Fingerprints, or client IP addresses. This shows that using the correct minimal headers the data can be accessed programmatically. The methodology and results demonstrate a systematic reverse-engineering approach that can be extended to more complex antibot systems.

Methodology

First, a search was made on Delta.com using an example flight (JFK→ATH, Oct 14, one way). Then using Web-Tools, set the filter Doc (Document) and got 4 html pages.

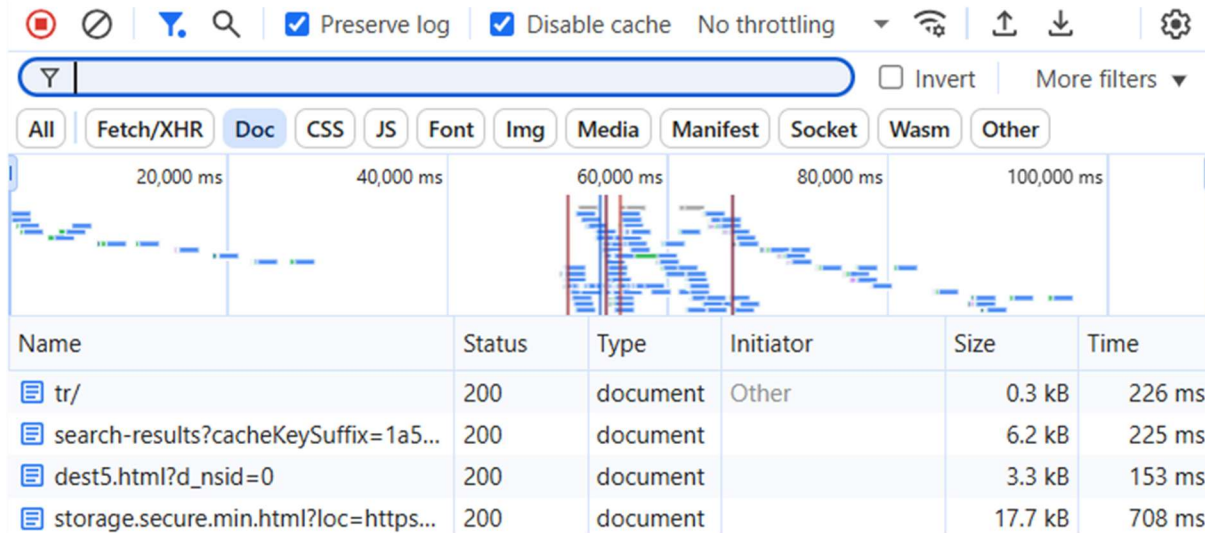
I dumped the two most relevant pages and searched through them for the API link. I then set the link as a filter along with Fetch/XHR.

Got the request to the API and tried to reproduce it on Linux with curl, and on PowerShell with Invoke-HttpRequest and httpClientRequest.



The screenshot shows the Delta flight search interface. At the top, it displays 'JFK' (New York-Kennedy, NY) and 'ATH' (Athens, Greece) with a double-headed arrow between them. To the right of the cities, it shows 'One Way' with a dropdown arrow, 'Oct 18' with a calendar icon, and '1 Passenger' with a dropdown arrow. A red 'SEARCH' button is on the far right. Below the search bar, there are 'SEARCH OPTIONS' with three checkboxes: 'Shop with Miles' (checked), 'Refundable Fares Only' (unchecked), and 'My dates are flexible' (unchecked). An 'Advanced Search' link with a dropdown arrow is also present.

Screenshot 1.1: Flight search



Screenshot 1.2: WebTools. Search request Document filter

Results

After setting up the exact same parameters, curl worked fine, returning the expected JSON (exactly as in the browser). However, in PowerShell, Invoke-HttpRequest did not work, returning a 444 error code. As soon as I switched to httpClientRequest, using the correct headers again, everything worked fine. More in-depth analysis can be found in Reports 1, 2, and 3, as well as in the dumps and scripts written throughout this project.

Screenshot 1.3: A flight offers JSON response

After reproducing the API request with curl, I tested the antibot defenses by systematically removing and altering headers, omitting cookies, switching between clients (curl, httpClientRequest, Invoke-HttpRequest), and sending requests from different IPs. These tests showed that Delta enforces specific headers and rejects chunked transfer encoding, while cookies, TLS/JA3 fingerprints, and IP changes did not affect the response. The detailed outcomes of each request are summarized in the table below.

4

airline: DL	Explicitly declares the airline. Likely required by Delta's backend API to scope results.	No
applicationid: DC	Identifies the calling application (Delta.com). Without it, you saw <i>RetailOfferError</i> .	Yes
authorization: GUEST	Authorization token (here, a guest session). Without it, the API returns <i>Unauthorized</i> .	Yes
cache-control: no-cache	Indicates the client wants fresh data, not cached. Common in browser traffic; helps mimic real behavior.	No
channelid: DCOM	Identifies traffic channel (Delta.com web). Missing it triggered errors.	Yes
content-type: application/json	Declares that the request body is JSON. Critical for POST/GraphQL APIs. If missing, the API wouldn't parse the payload.	Yes
cookies: dlsite=a; _fbp=fb.1.1758387998250.1...	Contains all session, tracking, Adobe/QuantumMetric analytics, and bot-detection cookies.	No
origin: https://www.delta.com	Declares the domain initiating the request. Enforced in CORS checks.	No
pragma: no-cache	Older HTTP directive similar to cache-control. Still sent by browsers for compatibility.	No
priority: u=1, i	HTTP/2+ priority hint header (used by Chrome). Makes the request look like real browser traffic.	No
referer: https://www.delta.com/	Shows the page where the request came from. Required	Yes

	— missing gives <i>Access Denied</i> .	
sec-ch-ua: "Chromium";v="140", "Not=A?Brand";v="24", "Google Chrome";v="140"	Client Hints — identify browser brand/version. Antibot requires them; removal gave <i>Access Denied</i> .	Yes
sec-ch-ua-mobile: ?0	Client Hint — whether browser is mobile. Required for realism.	Yes
sec-ch-ua-platform: "Windows"	Client Hint — OS platform. Required.	Yes
sec-fetch-dest: empty	Fetch metadata — indicates what the request is for (empty = API call). Missing = <i>Access Denied</i> .	Yes
sec-fetch-mode: cors	Fetch metadata — shows this was a CORS request.	No
sec-fetch-site: same-site	Fetch metadata — relationship between origin and request (same-site = delta.com → delta.com API).	No
transactionid: 9f5ca354-48a4-46bc-b958-4b82669dd446_1758510176242	Unique ID per request, used for tracking/logging. Missing it breaks the API.	Yes
user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36	Standard Chrome UA string. Helps requests look browser-like. While not always strictly enforced, antibots often flag unrealistic UAs.	Yes
x-app-type: shop-mach	Likely an internal application flag for "shopping machine" flows. Adds realism; may be validated lightly.	No

Table 1.1: Request parameters, their purpose and if they are enforced

```
curl 'https://offer-api-prd.delta.com/prd/rm-offer-gql' \
-H 'accept: application/json, text/plain, */*' \
-H 'accept-language: en-GB,en;q=0.9,el-GR;q=0.8,el;q=0.7,en-l' \
-H 'applicationid: DC' \
-H 'authorization: GUEST' \
-H 'channelid: DCOM' \
-H 'content-type: application/json' \
-H 'referer: https://www.delta.com/' \
-H 'sec-ch-ua: "Chromium";v="140", "Not=A?Brand";v="24", "Go' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Windows"' \
-H 'sec-fetch-dest: empty' \
-H 'transactionid: 5ae0375f-e4f4-4e60-92f6-5815d72aa711_1758' \
-H 'user-agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) App' \
--data-raw $'{"variables":{"offerSearchCriteria":{"productGr
```

Screenshot 1.4: The minimal working curl request

Findings

Based on all the tests run, the antibot does not check cookies, as the request works without the cookies header. Below is a list of common checks typically enforced by antibot systems, and whether Akamai enforces them in this case. The request works with `httpClientRequest` but not with `Invoke-HttpRequest` in PowerShell, due to the chunked format used by `Invoke-HttpRequest`, which many antibot systems reject. The fact that the request succeeds with both `curl` and `httpClientRequest` shows that no JA3 fingerprinting is enforced. The same request also worked from different IP addresses, indicating that the antibot is not IP-sensitive.

Test	Enforced
Akamai Challenge Cookies	No
TLS ClientHello/ JA3 fingerprint mismatch	No
Missing Incorrect ALPN or HTTP/2 Brotli Support	Yes
Header Inconsistencies	Yes
Request Flow (Direct API call)	No
IP reputation	Yes
Rate limiting/Parallel Requests	No

Authentication/Authorization/Unexpected Payload Shape	Yes
Replay/Reuse of Tokens across different Ips	No

Table 1.2: Common antibot tests and if they are enforced

What Ifs:

If the antibot enforced cookie checks, this would need to be addressed. One possible approach, without relying on a headless browser, would be to run a warm-up session on the main domain so the script could store the cookies in a cookie jar and then reuse them for the API request.

If the API enforced JA3/TLS fingerprint checks, the request would need to be constructed with a low-level client (for example, C with OpenSSL) that can precisely shape the ClientHello bytes to match a browser's JA3 fingerprint.

Conclusion

This research shows that Delta Airlines' antibot system (Akamai) relies primarily on the **integrity of request headers** and certain aspects of **client behavior** (e.g., rejecting chunked transfer encoding), rather than on Akamai-issued cookies. Additionally, no **JA3/TLS fingerprinting checks** were enforced, since both curl and httpClientRequest successfully returned valid JSON flight offers.

These findings demonstrate that the current defenses on this endpoint can be bypassed by replicating the correct header set with standard tools. However, if Delta strengthens its security in the future—such as by enforcing cookie validation, TLS fingerprinting, or behavioral checks—further measures will be required to adapt the request construction and maintain access.