Barkhausen AI

DATA INSIGHT · BA-DI-1

A comma in a User-agent line: a group no crawler matches

A robots.txt group begins with a User-agent line, and the value on that line is a product token — a name matched, case-insensitively, against a crawler's own token.

“Every leap begins as a crackle.”

DOCUMENT	BA-DI-1
KIND	Data insight
DATE	2026
SUBJECTS	Crawlers & access; Engines & documentation
LICENSE	CC-BY-4.0
AUTHOR	Barkhausen AI

ABSTRACT

A robots.txt group begins with a User-agent line, and the value on that line is a product token — a name matched, case-insensitively, against a crawler's own token. RFC 9309 fixes the characters a product token may contain: letters, hyphens, and underscores, and nothing else. A value carrying a comma is therefore not a product token, and a conformant parser matches no crawler to the group, so whatever Disallow rules follow never take effect. This note demonstrates the mechanism with the census's pinned parser on a constructed two-token line and a single-token control, then reports what the census corpus actually holds: ten news-sector robots files carry a comma-bearing User-agent value, none of them a crawler token — they are whole browser user-agent strings, a quoted bot description, and offline-downloader names with bracket flags — and not one file in the corpus joins two recognized crawler tokens with a comma. The finding is small and mechanical, but it separates a rule that is written from a rule that runs.

CONTENTS

1 The rule, demonstrated 3

2 What the corpus actually contains 3

3 Written is not the same as running 4

4 Limitations 4

A robots.txt file is read as a set of groups, and each group opens with one or more `User-agent` lines. The value on such a line is a *product token*: a name that a crawler matches, case-insensitively, against its own token to decide which group's rules it must obey. RFC 9309 does not leave the shape of that token open. Its grammar states that a product-token is an identifier built from a fixed character set — `%x2D`, `%x41-5A`, `%x5F`, `%x61-7A`, that is, the hyphen, the uppercase letters, the underscore, and the lowercase letters — and nothing else [1]. A comma is not in that set. Neither is a space, a slash, a digit, or a parenthesis. A `User-agent` value that contains any of them is, by the specification's own grammar, not a product token a crawler can match, and the group it heads is unreachable: its `Disallow` rules describe a restriction that never applies to anyone.

1 The rule, demonstrated

The census parses robots files with Protego 0.6.2, a pure-Python parser that follows Google's robots.txt implementation of RFC 9309 [2]. Run directly, it makes the consequence visible. The experiment below builds two robots texts — one whose `User-agent` line comma-joins two tokens, and a single-token control — and asks each named crawler whether it may fetch the site root:

```
from protego import Protego

comma  = "User-agent: GPTBot, ChatGPT-User\nDisallow: /\n"   # one comma-joined line
single = "User-agent: GPTBot\nDisallow: /\n"                  # control

rp_comma, rp_single = Protego.parse(comma), Protego.parse(single)
for agent in ("GPTBot", "ChatGPT-User"):
    print(agent, rp_comma.can_fetch("/", agent), rp_single.can_fetch("/", agent))

# GPTBot        True   False
# ChatGPT-User  True   True
```

In the control, the single-token group works exactly as written: `GPTBot` is denied the root (`can_fetch` is `False`), and `ChatGPT-User`, which the group does not name, is unaffected. In the comma-joined case, both crawlers are *allowed* the root. The parser treats `GPTBot, ChatGPT-User` as one opaque value, matches it against neither crawler's token, and finds no group that applies — so the default is to allow. A reader who expected the comma to mean "and" — block both — gets the opposite of the intent for both. The `Disallow: /` under that line is inert.

2 What the corpus actually contains

A line scan of the 1,381 parsed robots files in the 2026 crawler-access census finds ten domains — all in the news sector, none elsewhere — whose robots file carries a `User-agent` line with a comma in its value. It is worth being exact about what those ten lines are, because they are not the tidy two-crawler joins the mechanism above uses for illustration. Eight of the ten are near-identical files from one newspaper group, and their comma-bearing lines are entire browser user-agent strings pasted into the `User-agent` field — values of the form `Mozilla/5.0 (…; like Gecko) … (compatible; …)`, the comma sitting inside `(KHTML, like Gecko)`. One is a quoted, sentence-long description of a scraping bot. One lists offline-download tools followed by bracketed flags — `FrontPage [NC,OR]`, `HTTrack [NC,OR]` — a syntax borrowed from a different configuration language that robots.txt does not interpret. Across all four

sectors, no file joins two recognized crawler product tokens with a comma; the `GPTBot, ChatGPT-User` shape is a construction for demonstration, not an observed pattern.

The unifying property is what matters. In every one of the ten, the `User-agent` value contains characters the RFC's product-token grammar excludes, so none of them is a token any crawler will match. The authors were plainly trying to name *something* — specific scrapers, a browser-shaped signature, a pair of download tools — and in each case the group they wrote is one no conformant crawler enters. The intent varies; the outcome is uniform.

3 Written is not the same as running

This is a narrow observation, and it should be read narrowly: ten files, in one sector, is not a claim about how often site operators mis-address robots.txt in general. What it isolates is a gap between two things a census could count. A rule is *written* when it appears in a file; a rule *runs* when a crawler matching the group encounters it and obeys. The two are usually assumed to coincide, and for a well-formed `User-agent` line they do. A comma-bearing value is a small, verifiable case where they come apart — the rule is present in the bytes and absent from the crawler's behavior — and the same parser a study uses to measure "who blocks what" will, correctly, score these groups as blocking no one. The census reports its comma-bearing lines as their own signal for this reason: they are a reminder that a `Disallow` counts only against a `User-agent` line that can be matched. The full corpus and per-domain records are available in the crawler-access dataset.

4 Limitations

The comma is one way a `User-agent` value can fall outside the product-token grammar, not the only one; a value with a space, a slash, or a stray glyph is inert for the same reason, and this note counts only the comma case. The count of ten is a line-level scan of the parsed corpus and inherits that corpus's boundaries: files that could not be fetched or parsed are not examined, so this is a floor, not a population estimate. The demonstration uses the census's pinned parser, which implements Google's interpretation of RFC 9309; a crawler that departs from that interpretation — matching more leniently, or splitting on commas — could behave differently, and this note describes conformant parsing, not any particular operator's live behavior. Finally, the note reports what the ten values are, not what their authors intended, which can only be inferred from the strings themselves.

REFERENCES

1. M. Koster, G. Illyes, H. Zeller, and L. Sassman, IETF. *RFC 9309: Robots Exclusion Protocol* (2022). https://www.rfc-editor.org/rfc/rfc9309.html Accessed 2026-07-10. [archived]
2. Scrapy project. *Protego — a pure-Python robots.txt parser (v0.6.2)* (2026). https://pypi.org/project/Protego/ Accessed 2026-07-10. [archived]

HOW TO CITE

Barkhausen AI (2026). A comma in a User-agent line: a group no crawler matches. https://barkhausen.ai/notes/comma-in-user-agent-lines/