TECHNICAL CASE STUDY / AI SEARCH OPTIMIZATION

The Invisible AI Crawler Block

How the Sofitel Legend Metropole Hanoi passed every SEO test its team could run, while quietly returning an infinite redirect loop to GPTBot, ClaudeBot, PerplexityBot and Googlebot whenever they arrive through a proxy. One request header explains all of it.

SUBJECT
Sofitel Legend Metropole Hanoi

PLATFORM
WordPress, proxy-aware HTTPS rule

MEASURED
12 August 2026

STATUS
Reproducible, unfixed at origin

200 OK → 301 forever

Same URL. Same server. Same second. The only difference between a healthy page and an unreachable one is a single header that every proxy and CDN on the internet adds automatically.

Published by AI Crawler Check | aicrawlercheck.com
Research and analysis: Horatos | Brian Ho Marketing
English edition | Tiếng Việt available as a separate document
Published in the public interest. Method and findings are independently reproducible.

Executive summary

The Sofitel Legend Metropole Hanoi, a landmark five-star property operated under the Accor group, appeared by every test its marketing team could run to be perfectly healthy. The homepage loaded. `robots.txt` loaded. Third-party SEO crawlers reported no errors. Testing with an AI crawler user agent returned **200 OK**. There was nothing to find.

Our audit tool disagreed. It could not read the site at all, and it kept reporting an endless redirect. Two separate attempts to explain the contradiction were wrong, in opposite directions. The first blamed the hotel's server. The second retracted that and blamed our own infrastructure. Both were wrong for the same reason: nobody had isolated *what triggered* the difference.

Isolating it took a controlled experiment across seven request headers. The answer is that the origin server returns a permanent, self-referential 301 redirect to any request carrying the `X-Forwarded-Proto` header, and returns a normal 200 to any request without it. That header is not exotic. It is added automatically by essentially every reverse proxy, CDN and edge platform in existence, which means it is present on a large share of real AI crawler traffic and absent from almost every manual test a marketer would run.

> **THE FINDING IN ONE SENTENCE**
>
> The site is invisible to AI crawlers that arrive through a proxy, visible to everyone who tests it by hand, and the gap between those two facts is one header that no ordinary diagnostic tool sends.

Why this matters beyond one hotel

This failure mode has three properties that make it unusually dangerous, and all three are generalizable well past this one property:

1. **It is invisible to the standard toolkit.** Browsers, curl, Google's Rich Results Test and most SEO crawlers connect directly and send no `X-Forwarded-Proto`. Every one of them reports success.
2. **It produces no error signal anywhere.** There is no 404, no 500, no timeout and no blocked-bot warning. The server answers instantly and politely, with a redirect. Nothing appears in a broken-links report.
3. **It defeats retries.** A crawler that follows redirects will follow this one forever. We measured it still redirecting at hop 25. Crawlers do not treat that as a temporary fault to revisit later; they treat the URL as unreachable and move on.

The practical consequence is a site that cannot be cited by AI answer engines, with no diagnostic trail explaining why, and an owner with a folder full of green test results.

Part 1: The contradiction

The investigation began with a false diagnosis from our own tool. It reported **DNS resolution failed** for the hotel's domain. That was plainly wrong, since the domain resolves and the site loads in any browser, and

fixing that false report is what exposed everything underneath it.

Once the false DNS message was corrected, the tool reported the real failure: an endless redirect. The client tested it independently and reported back:

```
>>> https://www.sofitel-legend-metropole-hanoi.com
Status: 200 OK   Code: 200
X-Cache-Engine: WP-FFPC with predis via PHP
Vary: Accept-Encoding,User-Agent
# the client's conclusion: "not a 301 redirect issue"
```

The client was right about their measurement and wrong about the conclusion, and this is the single most instructive moment in the whole case. A 200 OK does not disprove a redirect loop when the loop is *conditional*. Their test was not a counterexample. It was, without either of us realizing it yet, the control arm of the experiment.

THE REASONING ERROR WORTH TEACHING

"I tested it and it works" is not evidence that a site works. It is evidence that the site works *under the exact conditions of that test*. When two competent parties get opposite results from the same URL, the correct conclusion is never that one of them is incompetent. It is that the request conditions differ, and the difference is the actual finding.

Part 2: Isolating the variable

The method here is ordinary experimental discipline, and it is worth spelling out because it is the transferable part of this case study. We held the URL, the server, the time window and the user agent constant, and varied exactly one request header at a time.

Requests to Cloudflare Workers carry a set of proxy headers automatically. The hypothesis was that one of them was the trigger. Rather than guess which, we sent each in isolation against the same path:

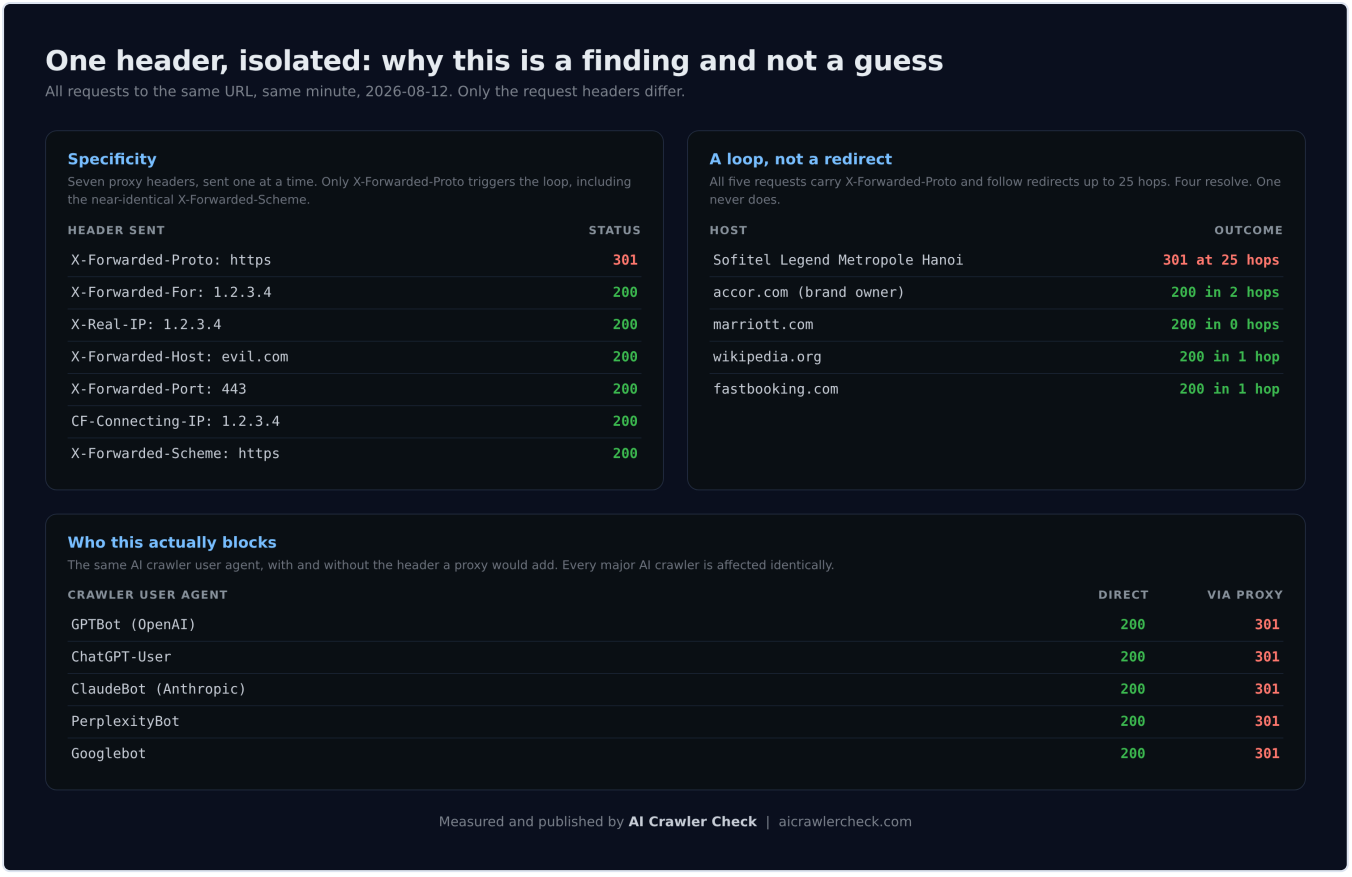

Figure 1. The complete measurement set, all against the same URL in the same minute. Left: seven proxy headers sent one at a time. Right: five hosts, all receiving the trigger header, following redirects to a limit of 25 hops. Bottom: five AI and search crawler user agents, tested with and without the header a proxy would add.

Result A: the trigger is specific

Only `X-Forwarded-Proto` produces the redirect. Six other proxy headers, including `X-Forwarded-Scheme`, which carries the identical meaning and differs only in name, all return a normal 200.

HEADER SENT (ONE AT A TIME)	STATUS
X-Forwarded-Proto: https	301
X-Forwarded-Scheme: https	200
X-Forwarded-For: 1.2.3.4	200
X-Real-IP: 1.2.3.4	200
X-Forwarded-Host: evil.com	200
X-Forwarded-Port: 443	200
CF-Connecting-IP: 1.2.3.4	200

That single contrast between `X-Forwarded-Proto` and `X-Forwarded-Scheme` is what elevates this from correlation to mechanism. A firewall or bot-detection system would not distinguish between two synony-

mous headers. A specific line of application code reading a specific variable name would, and does.

Result B: the value does not matter

Any non-empty value triggers the redirect, including values that make no sense in context. This tells us the server is testing for the header's *presence*, or comparing it incorrectly, rather than evaluating what it actually says.

VALUE SENT	STATUS	VALUE SENT	STATUS
`https`	301	`ssl`	301
`http`	301	`https,https`	301
`HTTPS`	301	`https;`	301
`on`	301	`(empty)`	200

Note the last row carefully. An empty value returns 200, but that is not a usable workaround, because a proxy platform that injects the header will not let a downstream application send it empty. We confirmed this directly: setting the value to an empty string in a Cloudflare Worker still delivered `https` to the destination server. The header is not removable from inside the platform that adds it.

Result C: it is a loop, not a redirect

A redirect is normal and harmless. A redirect that points at the URL just requested is neither. The `Location` header proves the self-reference:

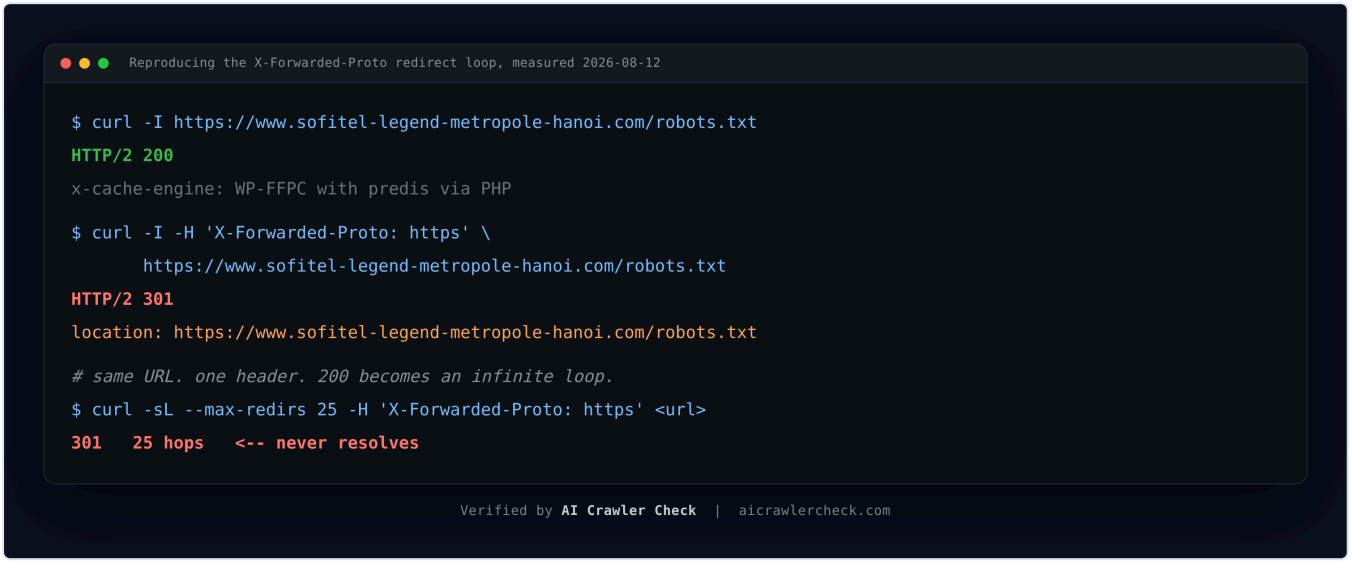

Figure 2. The reproduction, in two commands. The only difference is the `-H` flag. Note that the `location` value on the 301 is character-for-character the URL that was requested, which is what makes this a loop rather than a redirect. Following it reaches the 25-hop limit without ever resolving.

To confirm that the loop is genuinely abnormal rather than just a redirect chain we were impatient with, we sent the same header to four control hosts, including **the hotel brand's own parent company**. Every control resolves.

HOST (ALL SENT X-FORWARDED-PROTO: HTTPS)	OUTCOME
Sofitel Legend Metropole Hanoi	301 at 25 hops
accor.com (owns the Sofitel brand)	200 in 2 hops
marriott.com	200 in 0 hops
wikipedia.org	200 in 1 hop
fastbooking.com (hospitality booking platform)	200 in 1 hop

The accor.com row is the most useful line in this document for a client conversation. The parent company of the same hotel brand, serving the same industry, handles the identical header correctly. This is not an unavoidable cost of running a hotel website. It is a fixable configuration defect on one property.

Result D: the scope is the whole site

This is not limited to `robots.txt`. The homepage behaves identically: 200 without the header, 301 with it. Any AI crawler arriving through a proxy cannot read the robots policy *or* the content it governs.

Part 3: Who this actually blocks

The commercial impact depends entirely on one question: does real AI crawler traffic carry this header? For a meaningful share of it, yes. Large-scale crawlers rarely connect straight from a single machine. They run behind load balancers, egress proxies and CDN layers, and adding `X-Forwarded-Proto` is the standard, correct behavior for every one of those components. It is how a proxy tells the origin server which protocol the original visitor used.

We tested nine crawler identities both ways, spanning live AI retrieval, AI training corpora and conventional search indexing. The result is uniform:

CRAWLER USER AGENT	DIRECT CONNECTION	THROUGH A PROXY
GPTBot (OpenAI training)	200	301 loop
ChatGPT-User (live retrieval)	200	301 loop
ClaudeBot (Anthropic)	200	301 loop
PerplexityBot	200	301 loop
Google-Extended (Gemini training)	200	301 loop
Amazonbot	200	301 loop
Bytespider (ByteDance)	200	301 loop
Googlebot (search)	200	301 loop
Bingbot (search, Copilot)	200	301 loop

The user agent is irrelevant. That is important, because it rules out the explanation a marketing team would reach for first, which is deliberate bot blocking. Nobody configured this. No firewall rule names GPTBot. The server treats a browser and a crawler identically, and it is the transport path, not the identity of the visitor, that decides whether the site is readable.

WHY NOBODY CAUGHT THIS

Every diagnostic a hotel marketing team is likely to run connects directly: a browser, an online header checker, a robots.txt validator, Google Search Console's URL inspection, a desktop SEO crawler. None of them sit behind a proxy. None of them send the trigger. The tests were not sloppy. They were testing a path that works, and the failing path is one that no human ever manually travels.

Is Google affected, or only AI bots?

This is the first question a site owner asks, and the precise answer is more useful than a yes or a no. Nine crawler identities out of nine fail when the request passes through a proxy, and nine out of nine succeed when it does not. Search engine bots are not exempt. Googlebot and Bingbot behave exactly like GPTBot, because the server never looks at who is asking.

It does not follow that the site is invisible on Google. Three further measurements establish the boundary of the damage, and each one argues against overstating the finding:

1. **The origin is not behind a CDN.** The domain resolves directly to `lub-sg-1.wp-ha.fastbooking.com` and returns no CDN response headers. The redirect is generated by the WordPress origin itself, not by an intermediary sitting in front of it.

2. **The site is indexed.** A `site:` query returns the homepage together with deep pages including `/location/`, `/meetings-events/` and the Vietnamese subdirectory. Conventional organic visibility is intact, because Google predominantly crawls from its own machines with no intermediate proxy.
3. **The robots policy permits everything relevant.** It is a stock WordPress file: `Disallow: /wp-admin/`, one allow rule for `admin-ajax.php`, and a sitemap reference. No crawler is excluded by policy, and `sitemap_index.xml` is reachable and complete.

One measurement looked like further damage and is not. `/sitemap.xml` returns a 301, but it points to `/sitemap_index.xml` and resolves with a 200 in a single hop. That is ordinary Yoast SEO behavior, present on a large fraction of WordPress sites, and unrelated to the loop. We report it because ruling it out required testing it, and a case study that counts benign redirects as evidence is not worth reading.

THE SCOPE, STATED PRECISELY

Every crawler is affected, but only on requests that traverse a proxy, CDN or serverless edge layer. That is not a marginal path: it is where a large and growing share of AI retrieval traffic originates, and it is the one path no human ever travels by hand. The site works for the visitors its owner can measure and fails silently for the ones it cannot.

WHY THE NARROW SCOPE MAKES THIS MORE DANGEROUS, NOT LESS

A total block would have been found years ago. Search Console would report it, rankings would collapse, and someone would have escalated within a week. A defect that fires only on proxied requests produces no ranking signal, no console warning and no failed test, because every conventional diagnostic connects directly. That is precisely why it survived years of agency work and SEO audits which all correctly returned 200 OK. The narrower the failing path, the longer the failure lives.

The live audit result

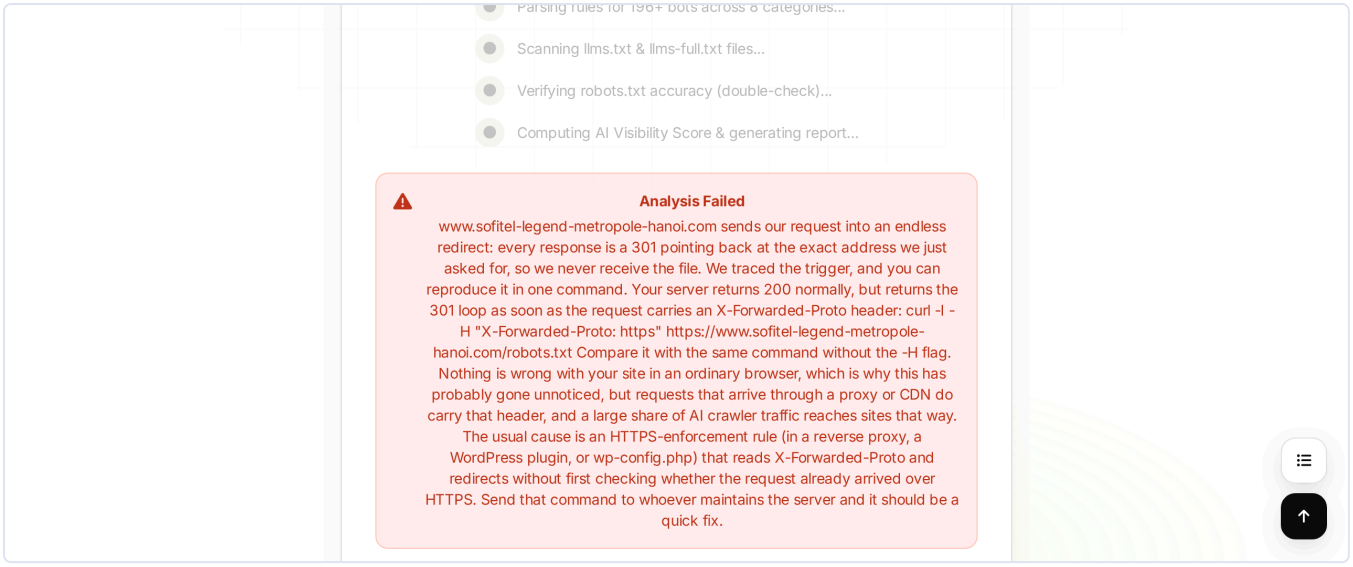

Figure 3. The production result at aicrawlercheck.com after this investigation. Rather than a generic failure, the report names the trigger header, hands over the exact command to reproduce it, states plainly that nothing is wrong in an ordinary browser, and identifies the likely configuration source. Diagnostics a site owner cannot act on are not diagnostics.

Part 4: The mechanism, and honest attribution

The server runs WordPress. The `x-cache-engine: WP-FFPC with predis via PHP` response header identifies the caching layer specifically. The behavior is the signature of an HTTPS-enforcement rule, and these are extremely common in three places: a reverse proxy configuration, a WordPress SSL plugin, or a snippet added to `wp-config.php`.

The defective logic looks approximately like this:

```
# the defect, in pseudocode
if (request has header X-Forwarded-Proto) {
    redirect_permanently(to: https version of this URL)
}
# the request was ALREADY https. so the "https version"
# is the same URL. so it redirects to itself. forever.
```

The rule was almost certainly written to solve a real problem: forcing HTTP visitors onto HTTPS when the origin server sits behind a proxy that terminates TLS. In that setup the origin genuinely cannot tell from the connection itself whether the visitor used HTTPS, so reading `X-Forwarded-Proto` is the correct approach. The bug is not in reading the header. The bug is redirecting without first checking whether the request already arrived over HTTPS.

ATTRIBUTION, STATED ACCURATELY

It is tempting to assign blame to one party. Neither framing survives the evidence. **The trigger comes from the visiting infrastructure:** proxy and edge platforms add `X-Forwarded-Proto` automatically, and a client running on such a platform cannot remove it. **The loop comes from the origin configuration:** the HTTPS-enforcement rule redirects unconditionally. Remove either half and the failure disappears. Only the origin half is fixable by anyone who has a stake in the site being visible.

The fix

The correction is small and low-risk. The enforcement rule must check the current protocol before redirecting, and must not redirect when the request already arrived over HTTPS:

```
# correct logic
if (X-Forwarded-Proto is present AND its value is not "https") {
    redirect_permanently(to: https version)
}
# already https? serve the page. do not redirect.
```

Whoever maintains the server can verify the fix with the same two commands used to find it. After the fix, both must return 200:

```
curl -I https://example.com/robots.txt
curl -I -H 'X-Forwarded-Proto: https' https://example.com/robots.txt
```

Part 5: What to take from this

For site owners

- **Test the path your visitors actually use, not the path that is convenient to test.** A browser is a direct connection. Much of your machine traffic is not.
- **A green result from a direct test tells you less than you think.** It confirms one path works. It says nothing about conditional behavior on other paths.
- **Audit HTTPS-enforcement rules specifically.** They are written once, usually years ago, usually by someone no longer involved, and they sit in front of every request forever.
- **Silent failures outrank loud ones in cost.** A 500 error gets fixed on the day it appears. A conditional redirect loop can suppress AI visibility indefinitely while rankings hold, Search Console stays quiet and every dashboard stays green.

For practitioners: the diagnostic method

TRANSFERABLE TECHNIQUE

1. **Treat contradictory results as data, not as error.** When two competent tests disagree, the disagreement itself localizes the bug.
2. **Vary one condition at a time.** Same URL, same second, same user agent, one header changed. Anything less does not isolate a cause.
3. **Include a near-miss control.** Testing `X-Forwarded-Scheme` alongside `X-Forwarded-Proto` is what ruled out "the server dislikes proxy headers" and proved that one specific variable name is being read.
4. **Include a peer control.** Sending the same header to accor.com, marriott.com and wikipedia.org established that correct handling is normal and this behavior is not.
5. **Distinguish "responds" from "resolves."** A server answering instantly with a redirect is not a working server. Always follow the chain and count the hops.
6. **Verify your own claims before publishing them.** This investigation produced two confident, wrong diagnoses before the correct one. Both were retracted in writing. Diagnostic authority comes from the willingness to be corrected by measurement.

For AI search visibility work generally

The broader lesson is about where AI visibility is actually won and lost. A great deal of published GEO advice concerns content structure: schema markup, answer formatting, entity clarity, citation-friendly phrasing. All of that matters, and all of it is irrelevant if the crawler receives a 301 loop instead of the document.

Technical reachability is not a prerequisite that gets checked once and forgotten. It is a live surface that can break silently, conditionally, and without leaving a trace in any standard report. Verifying it is the first job, not a formality preceding the real work.

REPRODUCE THIS YOURSELF

Every measurement in this document was produced with `curl`, which ships with macOS and Linux and is available on Windows. Substitute any domain you want to audit:

```
curl -I https://YOURSITE.com/robots.txt
curl -I -H 'X-Forwarded-Proto: https' https://YOURSITE.com/robots.txt
# both should return 200. if the second returns 3xx
# pointing at the same URL, you have this bug.
```

All measurements in this case study were taken on 12 August 2026 against the live production site and were repeated to confirm stability. The findings are reproducible by any third party using the commands shown. The site was not altered, probed for vulnerabilities, or accessed beyond ordinary public HTTP requests. The property is named because the evidence is only useful if it can be verified, every command in this document can be run by any reader against the public site, and the defect is a configuration error rather than a security weakness. Analysis performed with AI Crawler Check, which tracks crawler access policy for 248 bots across 8 categories, 177 of them named in their operator's own documentation.