AIHACKERS Sample *Report*

External surface assessment · Example document, fictitious client

Client · **Grupo Ejemplo, S.A. de C.V.**

Version · **1.0, sample**

Classification · **Public, commercial**

Client: Grupo Ejemplo, S.A. de C.V. **(fictitious company)**
Target: `grupo-ejemplo.example` **(reserved domain, does not exist)**
Profile: 240 employees · 34 public hostnames · 4 cloud providers
Version: 1.0, sample
Classification: Public, commercial

> ## This document is a sample
>
> **It does not correspond to any real assessment or any real client.** The company, the domain, the hostnames and every IP address are fictitious: the domain uses the `.example` TLD, reserved by RFC 2606 and unable to resolve, and the addresses come from the RFC 5737 documentation ranges. None of the systems described here exist.
>
> What is real is **the shape**: the structure, the evidence quoted verbatim, the discipline with which each finding is rated, remediation down to the configuration line, and the way what was not tested is declared. The findings reproduce the patterns this practice encounters most often, so that an organisation with a similar profile will recognise its own.

How to read this report

Four things, so you do not have to read all of it:

1. **§3 is the executive summary.** One page: root cause, headline finding and what to do first.
2. **Every finding has two boxes.** The coloured one at the top states the risk in one line. The green one at the bottom says what to do. If you only read those two per finding, you have 80 percent.
3. **§9 states what was NOT tested**, and why. An assessment without declared limits is not verifiable.
4. **§8 documents a finding that was withdrawn.** It is there on purpose.

---

1. Metadata

FIELD	VALUE
Client	Grupo Ejemplo, S.A. de C.V. (fictitious)
Target	`grupo-ejemplo.example` and the subdomains that resolve under it
Surface	34 hostnames, 19 IP addresses, 4 providers
Type	External black box, authorised, **non-destructive**
Authorisation	Scope closed in writing, signed by the client
Methodology	iNBest CVP
Standards	OWASP WSTG v4.2 · OWASP API Security Top 10 2023 · PTES · NIST SP 800-115
Evidence	58 files, index in §10
Prepared by	AIHACKERS · iNBest CVP

2. How it was run, and what it covered

The assessment runs a fixed checklist. It is not discretionary: every step leaves an evidence file, and a quality gate blocks delivery of the report if a step produced no evidence to support its result.

#	STEP	STATUS	EVIDENCE
1	Subdomain enumeration, 5 independent sources	Run	`all_hostnames_merged.txt`
2	Service inventory from device search engines	Run	`shodan_inventory.json`
3	Historical passive DNS	Run	`virustotal.json`
4	Port and banner scan	Run	`nmap_live_ips.txt`
5	Email and DNS authentication	Run	`email_auth.json`
6	TLS and certificate posture	Run	`cert_audit.json`
7	Security headers	Run	`headers_audit.json`
8	Subdomain takeover and dangling records	Run	`takeover.json`
9	Out-of-support software	Run	`stack_eol.json`
10	Bundle and source map review	Run	`sourcemap.json`
11	Credentials in breach corpora	Run	`breach_corpus.xlsx`
12	Pastes and public repositories	Run	`paste_recon.json`
13	Exposed API documentation	Run	`api_docs.json`
14	Endpoint authentication matrix	**Partial**	See §9
15	Vulnerability template scanning	Run	`nuclei.jsonl`
16	Evidence quality gate	**Passed**	zero exit

About step 1. Five sources are used, not one, because none is complete. In this assessment the Certificate Transparency logs saw **11 of the 34** names, since much of the estate is served under wildcard certificates. The resolver sweep contributed 9 names no other source had, and **the exposed source map contributed one that none of the five found** (CRIT-A).

3. Executive summary

Overall risk: High.

Root cause. The four highest-severity findings share one pattern: **artefacts that reached the Internet without anyone deciding to publish them.** A source map inside the production build, four

non-production environments with public names, an administrative console on its default path, and a redirect that downgrades encryption. None of them is a software vulnerability. All four are deployment decisions nobody reviewed.

There is a second, subtler pattern: **controls that exist but do not apply.** DMARC is published in monitoring mode, so it blocks nothing. The content policy is extensive but allows forms to submit to any destination. Both read as protection in a configuration audit and stop nothing in practice.

The headline finding. The partner portal was published with its source map. That file handed over the original code of 28 files, the full contract of the internal API, and the name of a server that **none of the five enumeration sources had found**. An attacker enumerating subdomains the usual way does not see it; anyone who downloads the map does.

Posture.

SEVERITY	COUNT
Critical	1
High	3
Medium	5
Low	2
Informational	1

What to do first. The first two are one-line changes:

1. Delete two files from storage and turn off their generation in the build (CRIT-A).
2. Change the target of a redirect and publish a header (HIGH-C).
3. Reset 19 identities and revoke their sessions (HIGH-B).

What is done well. DNS zone transfer is correctly refused on all four name servers. The store's admin panel uses a non-default name, which is good practice applied deliberately. There are no domains open to takeover. It is said because a report that only lists defects is not an assessment, it is a list of complaints.

---

4. How severity is rated

CVSS is not used. CVSS scores the vulnerability in the abstract; this rubric scores **the client's real exposure**. A finding does not move up a category because it took effort to find.

Floors, never rated below:

SITUATION	FLOOR
Readable source code or secret (source map, published `.env`, embedded token)	**High**, even if the secret is never used
Unauthenticated access to personal or customer data	**High**
Software more than 4 years out of support, or a version with a pre-authentication CVE	**High**

Ceilings, never rated above:

SITUATION	CEILING
Version disclosure with no known vulnerable version	**Informational**
A vulnerability whose precondition is a compromise that was not demonstrated	**Medium**

That last ceiling is what keeps the report honest. If a finding needs a script injection that was not found, the real exposure is rated, not the hypothetical chain.

5. Surface inventory

HOST	FUNCTION	PROVIDER	OBSERVED
`www`	Corporate site	CDN	301 to `http://` (HIGH-C)
`tienda`	Online store	Public cloud	Commerce platform with integrated payments
`portal`	Partner portal	Static storage + CDN	**Public source map (CRIT-A)**
`api-datos`	Internal API	Public cloud	**Not enumerable; revealed by the map**
`erp`	ERP integration	Own data centre	Console on default path (MED-C)
`soporte`	Support portal	Public cloud	Expired certificate (MED-D)
`qa`, `dev`, `stage`, `uat`	Non-production	Public cloud	Public (MED-A)
`vpn`, `correo`	Remote access and email	Own data centre	

Coverage by source:

```
SOURCE                          NAMES     UNIQUE TO THAT SOURCE
Certificate Transparency        11 of 34                    3
Device search engine            22 of 34                    6
Historical passive DNS          18 of 34                    4
Third-party corpus              14 of 34                    1
Resolver sweep, 557 labels      21 of 34                    9
Exposed source map               1 of 34                    1   <-- CRIT-A
```

6. Detailed findings

If you only read two things per finding, read the coloured box at the top and the green box at the bottom.

CRIT-A Production source map exposes the partner portal and a non-enumerable server

ASSET `portal.grupo-ejemplo.example` **SEVERITY** Critical **STATUS** Open

THE RISK, IN ONE LINE

The partner portal's code is public, and it hands over the full map of an internal API that appears in no subdomain inventory.

What we found. The portal is a single-page application published on static storage behind a content delivery network. It was deployed with its source map, and the map contains `sourcesContent`, meaning the complete original code and not just the list of file names.

REPRODUCTION · READ-ONLY, NO AUTHENTICATION

```
# 1. The compiled bundle declares its own map on the last line
curl -s https://portal.grupo-ejemplo.example/static/js/main.<hash>.js | tail -c 120

# 2. The map answers
curl -sI https://portal.grupo-ejemplo.example/static/js/main.<hash>.js.map
```

SERVER RESPONSE · WHAT IT HANDS ANYONE

```
//# sourceMappingURL=main.<hash>.js.map

HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 4518621
Cache-Control: public, max-age=31536000

{"version":3,"sources":[...612 paths...],"sourcesContent":[...original code...]}
```

612 sources, of which **28 are the application's own files** and the rest are libraries. The 28 are recovered verbatim:

OWN FILES RECOVERED FROM THE MAP

```
services/Service.jsx                components/auth/SignIn.jsx
utils/Constants.js                  components/auth/ForgotPassword.jsx
components/pages/ClientesView.jsx   components/pages/RegistrosView.jsx
components/pages/DetalleView.jsx    components/pages/Uploader.jsx
```

Reading two of them hands over the full API contract:

API CONTRACT, RECOVERED FROM `UTILS/CONSTANTS.JS`

```
export const API_URL = process.env.REACT_APP_API_URL;

export const SIGN_IN_URL       = `${API_URL}/api/v1/sign-in`;
export const RESET_OTP_URL     = `${API_URL}/api/v1/resetPasswordWithOtp`;
export const GET_CLIENTES_URL  = `${API_URL}/api/v1/getClientes`;
export const GET_REGISTROS_URL = `${API_URL}/api/v1/getRegistros`;
export const GET_REPORTE_CSV   = `${API_URL}/api/v1/getReporteCSV`;
export const UPLOAD_URL        = `${API_URL}/api/v1/cargarCatalogo`;
```

`API_URL` is an environment variable in the code, but the production build writes it literally into the bundle:

SERVER ADDRESS, INLINED IN THE COMPILED BUNDLE

```
const bt="https://api-datos.grupo-ejemplo.example",
      vt="".concat(bt,"/api/v1/sign-in")
```

Why it matters. `api-datos` did not appear in any of the five enumeration sources: not Certificate Transparency, not the device search engine, not passive DNS, not the third-party corpus, not a 557-label dictionary sweep. **The source map is the only reason it is in this report.** An attacker enumerating the usual way does not find it; one who downloads the map does.

The recovered endpoints include a bulk CSV export and a query that takes the customer identifier as a browser parameter (see HIGH-A).

What we verified and what we did not. We confirmed that the map answers, that it contains the original code, and that the bundle names that server. **No active credential was found** in the map or the bundle: the strings the tool flagged as possible secrets turned out, on review one by one, to be constants from third-party libraries. It is said because overselling a finding serves no one. The severity rests on the code disclosure and on the server it uncovered, not on a leaked key.

REMEDIATION

1. **Delete the `.map` files from storage and invalidate the content delivery network cache.** Until the cache is invalidated, the file keeps being served even though it no longer exists at the origin.
2. Set the variable in the production build so it does not come back on the next deployment: `GENERATE_SOURCEMAP=false`
3. Keep the maps where they are useful: as a continuous integration artefact, or uploaded to the error tracking service, which consumes them privately.
4. Treat `api-datos` as a name already known to third parties. Removing it from DNS does not protect
   it: it would still be reachable by IP address.

HIGH-A The records query accepts the identifier supplied by the browser

ASSET `api-datos.grupo-ejemplo.example` SEVERITY High STATUS Open

THE RISK, IN ONE LINE

If the server does not validate who owns the record being queried, one partner could read another partner's customer data.

What we found. The code recovered in CRIT-A shows how the portal builds the query and how it carries the session.

EXCERPT OF `SERVICES/SERVICE.JSX`, RECOVERED FROM THE MAP

```
export async function getRegistros(page, limit, clienteId) {
    return fetch(
        `${GET_REGISTROS_URL}?clienteId=${clienteId}&page=${page}&limit=${limit}`,
        { method: "GET",
          headers: { Authorization: localStorage.getItem("sessionToken") } }
    ).then((r) => r.json());
}
```

Two weaknesses visible in the code itself:

- `clienteId` travels as a browser parameter. The parameter coming from the client is not itself the defect; the defect would be the server not verifying ownership.
- The session token lives in `localStorage`, readable by any script on the origin, and is sent without a scheme prefix. A script injection in the portal becomes full session theft instead of a contained defect.

Why it matters. It maps to **API1:2023, Broken Object Level Authorization** in the OWASP API Security Top 10, which tops that list for both frequency and impact.

This is a precondition, not a confirmed vulnerability. Determining whether the server validates ownership requires authenticating with two different partner accounts and checking whether one reaches the other's records. That is authenticated testing and was outside the agreed scope. It is documented because the CRIT-A disclosure makes the endpoint and the parameter name public, so the precondition went from theoretical to known.

REMEDIATION

1. **Derive the partner's scope on the server** from the authenticated session and check it against the requested `clienteId` on every call. Do not trust the browser parameter.
2. Move the token to a cookie with all three attributes, and send it with a scheme: `Set-Cookie: session=<token>; HttpOnly; Secure; SameSite=Lax; Path=/ Authorization: Bearer <token>`
3. **Authorise an authenticated test across two accounts** to settle the question. It is the highest-value pending check in this report, and it takes less than an hour.

HIGH-B Plain-text corporate credentials in public corpora

ASSET Corporate domain **SEVERITY** High **STATUS** Open

THE RISK, IN ONE LINE

19 corporate passwords circulate publicly in plain text, and they are reusable against remote access and email.

What we found. 84 records with a corporate email address in public credential corpora. The composition matters more than the total:

CORPUS COMPOSITION, BY ORIGIN

ORIGIN	RECORDS	IN PLAIN TEXT
Combo lists derived from malware	23	19
Enterprise data broker	41	0
Historical breaches of third-party sites	20	0
TOTAL	84	19

Why the number that counts is 19. The 41 broker records are professional contact information: name, email, job title. They contain no credential. The 20 from historical breaches carry a hash, not

a usable password.

The real rotation scope is 19 identities, not 84. The distinction is not cosmetic: asking an IT team to reset 84 accounts when 65 have no compromised credential spends their time and the credibility of whoever asked. The next time something is reported, they will weigh it differently.

About age. The source corpora mix old lists with malware logs that are not recent either. The defensible claim is that **plain-text credentials exist in public circulation**, not that they were stolen recently. Age does not neutralise them: an unrotated password stays valid indefinitely, and reuse across services is the norm.

The passwords are not reproduced in this document and were not used against any system.

REMEDIATION

1. **Reset the 19 identities with a plain-text password AND revoke their sessions and refresh tokens.** A reset on its own does not invalidate a session cookie that was already stolen: the attacker stays in until the session expires on its own.
2. Confirm that the second factor is enforced across the whole tenant, preferably phishing-resistant (FIDO2 or certificate) and not OTP, which malware logs also capture.
3. Treat the remaining 65 as a known phishing target list: that is handled with targeted awareness, not password rotation.
4. Set up continuous monitoring of these corpora. They are updated constantly.

HIGH-C Every visit to the site passes once over clear-text HTTP, and there is no HSTS

ASSET `www` and the remaining 3 web hosts **SEVERITY** High **STATUS** Open

THE RISK, IN ONE LINE

The canonical redirect drops from HTTPS to HTTP before recovering, and no host tells the browser to refuse that downgrade, on a domain that processes payments.

REPRODUCTION

```
curl -sI https://www.grupo-ejemplo.example/ | grep -i '^HTTP\|^location'

for h in www tienda portal soporte; do
  echo -n "$h: "
  curl -sI https://$h.grupo-ejemplo.example/ | grep -ci strict-transport-security
done
```

SERVER RESPONSE

```
HTTP/1.1 301 Moved Permanently
location: http://grupo-ejemplo.example/        <-- scheme downgrade

www: 0
tienda: 0
portal: 0
soporte: 0
```

The full chain is `https://www` → **`http://apex`** → `https://apex`. The browser recovers encryption on the second hop, but the request has already travelled once in clear text.

Why it matters. An attacker on the network path (a public wireless network, a compromised provider hop, a home router) sees that hop and can strip encryption from the rest of the session. HSTS is the control designed precisely to make the browser refuse, and it is not published on any of the four hosts.

REMEDIATION

1. **Fix the redirect target so it keeps the scheme.** One line:
   `nginx return 301 https://grupo-ejemplo.example$request_uri;`
2. Publish the header on the apex and on `www`:
   `nginx add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;`
   Deploy `includeSubDomains` **only after** confirming that every subdomain serves HTTPS correctly, including the non-production ones in MED-A.
3. Once stable for a few weeks, consider inclusion in the HSTS preload list.

MED-A Four non-production environments with public names

ASSET `qa`, `dev`, `stage`, `uat` SEVERITY Medium STATUS Open

THE RISK, IN ONE LINE

Non-production environments run older versions, show detailed errors and often hold copies of real data.

OBSERVED STATE OF EACH ENVIRONMENT

HOST	HTTP	SERVER	OBSERVATION
stage	200	<server>/1.18.0	older version than production
dev	200	<server>/1.24.0	application error trace visible
qa	200	<server>/1.24.0	no edge authentication
uat	503	load balancer	no active targets

Three of the four were invisible to Certificate Transparency and appeared only in the resolver sweep. `uat` points to a load balancer with no targets: it is cost without function, and it becomes a takeover risk the day that load balancer is deleted and the DNS record is left pointing at nothing.

Why it matters. `stage` runs an older server version than production, which shows the patch cycle does not reach non-production environments. And `dev` returns the application trace, which reveals internal paths, library versions and sometimes query fragments.

REMEDIATION

1. **Put all four behind a VPN or an allow list of addresses.** None of them needs to be reachable from the open Internet.
2. Turn off development error display on any environment with public DNS.
3. Bring non-production environments into the same patch cycle as production, or decommission them.
4. Delete the `uat` DNS record and add record removal to the decommissioning procedure.
5. Issue wildcard certificates for non-production, so individual names stop being published in Certificate Transparency.

MED-B

Email spoofing of the domain is not blocked

ASSET Corporate domain SEVERITY Medium STATUS Open

THE RISK, IN ONE LINE

A spoofed email in the company's name is delivered instead of rejected, to partners and customers who expect quotes and invoices.

OBSERVED DNS RECORDS

```
$ dig +short TXT grupo-ejemplo.example
"v=spf1 include:_spf.<mail-provider> ip4:198.51.100.0/24 ~all"
                                                          ^^^^  softfail

$ dig +short TXT _dmarc.grupo-ejemplo.example
"v=DMARC1; p=none;"
               ^^^^ monitoring only, and without rua= nothing is collected

DKIM    no selector found among the common names
CAA     absent
DNSSEC  absent (no DNSKEY, no DS)
```

Why it matters. The three controls fail in different, complementary ways:

- **SPF ends in `~all`.** A message from an unlisted sender is marked suspicious, not rejected. It is still delivered.
- **DMARC at `p=none`** tells the receiver to **take no action** on messages that fail. And without `rua=` the aggregate reports are not collected either, and they are what would show who is spoofing the domain today.
- **Without DKIM**, much of the legitimate mail has no cryptographic signature to align against.

Without CAA, any public certificate authority can issue a certificate for the domain. It is the cheapest control in the report and it is not in place.

For a company whose business runs on quotes, purchase orders and invoices, this is the direct enabler of email fraud in its own name, against its own customers. It is the finding with the shortest path to a financial loss.

REMEDIATION

1. **Publish DKIM** for the mail provider and confirm it aligns with the visible domain.
2. **Add a report address to DMARC**, read the aggregates for a few weeks, and step up:
   `_dmarc TXT "v=DMARC1; p=none; rua=mailto:dmarc@grupo-ejemplo.example; pct=100" -> "v=DMARC1; p=quarantine; rua=...; pct=25" (then 50, then 100) -> "v=DMARC1; p=reject; rua=..."`
3. **Change SPF to `-all`** once the sender inventory is confirmed complete.
4. **Publish CAA** restricted to the authorities in use:
   `@ CAA 0 issue "<ca-in-use>" @ CAA 0 iodef "mailto:seguridad@grupo-ejemplo.example"`

MED-C Administrative console reachable on its default path

ASSET `erp.grupo-ejemplo.example` SEVERITY Medium STATUS Open

THE RISK, IN ONE LINE

The administration interface of the ERP integration answers from the open Internet, with no source restriction.

What we found. The console answers and presents a login screen.

OBSERVED RESPONSE

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Set-Cookie: <session>; path=/; HttpOnly
X-Frame-Options: SAMEORIGIN

<title>Panel de administracion</title>
```

What is confirmed and what is not. It is confirmed that the console is reachable from any Internet address. **No login was attempted**, neither with the HIGH-B credentials nor by any other means: that is authenticated testing and was out of scope. Nothing is claimed about the strength of the authentication, because it was not tested.

A contrast worth recording: the store's admin panel **does** use a non-default path, and returns 404 on the original one. The correct practice already exists inside the organisation; it just was not applied here.

REMEDIATION

1. **Restrict the console by IP address or VPN.** It is the control that closes the finding: `nginx location /admin { allow 198.51.100.0/24; deny all; }`
2. Change the default path, repeating what was already done on the store.
3. Require a phishing-resistant second factor on that interface.

MED-D Expired certificate on the support portal

ASSET `soporte.grupo-ejemplo.example` SEVERITY Medium STATUS Open

THE RISK, IN ONE LINE

The portal the support staff use presents an expired certificate, which trains the team to ignore browser warnings.

CERTIFICATE STATUS

```
subject   CN=soporte.grupo-ejemplo.example
issuer    <certificate authority>
notAfter  ~4 months ago            <-- EXPIRED
chain     valid, name matches
```

Why it matters, and it is not cryptographic. Encryption still works. The damage is behavioural: a team used to clicking "continue anyway" several times a day loses the ability to tell that warning apart from the one a real interception attack produces. An organisation's most expensive security control is its people's attention, and this spends it.

REMEDIATION

1. Renew the certificate and **automate renewal**. Automating it is what prevents a repeat; renewing it by hand guarantees it happens again.
2. Add expiry monitoring with a 30-day alert on every certificate in the estate, not just this one.

MED-E Permissive content policy on the payment path

ASSET `tienda.grupo-ejemplo.example` SEVERITY Medium STATUS Open

THE RISK, IN ONE LINE

The policy allows forms to submit to any destination, which is exactly the directive that would stop card data from being exfiltrated.

RELEVANT DIRECTIVES OF THE PUBLISHED POLICY

```
script-src       ... 'unsafe-inline' 'unsafe-eval'
default-src      ... 'self' 'unsafe-inline' 'unsafe-eval'
form-action      ... *                        <-- wildcard
frame-ancestors  *.<extension-provider> 'self'
```

Why it matters. The policy runs to several kilobytes and reads as a robust control. Three directives hollow it out:

- `'unsafe-inline'` with `'unsafe-eval'` in `script-src` means the policy **cannot stop injected code**, which is the main reason a content policy exists.
- `form-action` **with a wildcard** lets a form on the page send data to any destination. On a payment path, that removes the only directive that would block a client-side card skimmer.

- `frame-ancestors` authorises an external extension provider to embed the store in a frame.

REMEDIATION

1. **Remove the wildcard from** `form-action` and set it to `'self'` plus the named payment gateways.
   It is the highest-impact and cheapest change of the three.
2. Remove the external provider from `frame-ancestors` unless an active integration requires it.
3. Work towards removing `'unsafe-inline'` through per-request nonces. It is the most laborious, so
   it is best sequenced after the other two.

LOW-A Disclosure of versions and internal infrastructure

ASSET 4 web hosts **SEVERITY** Low **STATUS** Open

THE RISK, IN ONE LINE

Minor on its own; taken together it hands over a precise target profile, for free.

OBSERVED DISCLOSURE

```
Server: <server>/1.24.0 (<distribution>)      exact version, 4 hosts
X-Host: <internal-production-name>            internal server name
/version                        HTTP 200      version endpoint reachable
Content-Security-Policy                       lists every integrated provider

MISSING on all 4 hosts: Referrer-Policy, Permissions-Policy
```

The content policy also lists every integrated external provider. It is not a defect in itself, but an attacker gets from a single header the complete list of third parties to check for security advisories.

REMEDIATION

1. Turn off version publication and remove the internal header:
   `nginx server_tokens off; proxy_hide_header X-Host;`
2. Return 404 on the version endpoint.
3. Add `Referrer-Policy: strict-origin-when-cross-origin` and a minimal `Permissions-Policy`.
4. Trim the content policy to the origins actually in use.

LOW-B DNS records pointing to hosts that no longer answer

ASSET Domain zone **SEVERITY** Low **STATUS** Open

THE RISK, IN ONE LINE

Eleven records point to addresses that no longer answer. None can be taken over today, but one could be if the address is released.

Each of the eleven was checked against the regional address registry. **Zero takeover candidates** at the time of the assessment: the addresses are still assigned to the organisation or its provider.

The risk is latent and time-dependent. The day a provider releases one of those addresses and reassigns it to another customer, the organisation's DNS record will point to a third party's infrastructure.

REMEDIATION

1. Delete the eleven records.
2. **Add DNS record removal to the service decommissioning procedure.** It is the measure that stops
   them from piling up again.

INFO-A Clean results, recorded

ASSET Entire estate **SEVERITY** Informational **STATUS** Open

THE RISK, IN ONE LINE

None. It is documented so that a later assessment does not repeat the work, and to record what is done well.

- **DNS zone transfer refused** on all four name servers. Tested against all four, not one.
- **No subdomain takeover candidates**, including delegations to external marketing platforms, which are the ones that usually fail.
- **Store admin panel on a non-default name**, and 404 on the original path.
- **No exposed package registry or artefact store**: 23 candidate names tested, none resolves.
- **Sensitive platform paths correctly absent**: configuration files, version control directory and error logs return 404.

- **No detections** in the reputation services queried, and no malware sample observed communicating with the domain.

> **REMEDIATION**
>
> 1. No action required. Keep it that way, and verify in the next assessment that it still holds.

---

7. Narrative: how one file led to an invisible server

The full chain is worth laying out, because it explains why the headline finding is not produced by a scanner.

Step 1. The resolver sweep finds `portal`, a name Certificate Transparency did not have because the estate uses wildcard certificates.

Step 2. The portal turns out to be a single-page application served from static storage. That pattern is the signal: builds of that kind of application generate a source map, and it is often published by default.

Step 3. The map is requested. It answers 200, at 4.5 MB.

Step 4. The map includes `sourcesContent`. It is not a list of names, it is the original code.

Step 5. Two files in the code hand over the full API contract. The build also writes the backend server's address literally.

Step 6. That server was not in any of the five enumeration sources.

No step required an offensive technique. A file was requested and read. What changes the outcome is not sophistication, it is knowing that it had to be asked for.

---

8. A withdrawn finding, and why it is documented

The first pass reported the name servers as open recursive resolvers, a Medium finding with amplification risk.

Before publishing it, the mandatory control test was run: the same query against an address **out of scope and not owned by the client**.

CONTROL TEST THAT INVALIDATED THE FINDING

```
$ dig @<client-server>         ejemplo-externo.test  +short
  recursive answer      -> apparent open resolver

$ dig @192.0.2.1               ejemplo-externo.test  +short   # control address,
  recursive answer      -> SAME RESULT                         # not owned by the client
```

A documentation address that hosts no server cannot resolve anything. That it returned the same result shows the measurement was contaminated at the point of origin, not that the client had an open resolver.

The finding was withdrawn completely, and this section is the record of it.

It is included in a commercial sample on purpose. A provider that never withdraws anything is not being rigorous, it is being convenient. And the discipline that forces this one to be deleted is the same one that makes the other twelve credible.

---

9. What this assessment did not cover

It is declared because an assessment without declared limits is not verifiable.

Out of scope by agreement:

- **Authenticated testing.** Nothing behind a login. It is what HIGH-A needs in order to close, and what prevented assessing the strength of the authentication in MED-C.
- **Exploit validation.** No finding was taken beyond confirming that the condition exists. No data was accessed and no privileges were escalated.
- **State-changing actions**, load testing and denial of service.
- **Social engineering** and phishing against staff.
- **Source code review**, beyond what the exposed map handed over on its own.
- **Third-party assets**, including the providers the content policy names. Their vulnerabilities are reported to the client; fixing them is the provider's job.

Collection limits, recorded as unknown and not as clean:

- Step 14, the endpoint authentication matrix, ran **partially**: it covers the endpoints reachable without a credential. The part that requires a credential remains pending.
- The credential corpus returns a subset of the total available. **The figures are a floor.**
- Certificate Transparency saw 11 of 34 names. The other four sources covered the difference, which is why five are used and not one.

---

10. Evidence index

58 files support this report. Each finding cites its own.

FILE	CONTENT
`all_hostnames_merged.txt`	34 names consolidated from 5 sources
`shodan_inventory.json`	Service inventory by address
`nmap_live_ips.txt`	Ports and banners of the 19 addresses
`headers_audit.json`	Security headers per host
`cert_audit.json`	Validity, chain and name match
`email_auth.json`	SPF, DKIM, DMARC, DNSSEC, CAA matrix
`sourcemap.json`	Reachable maps and reconstructable sources
`main.<hash>.js.map`	The exposed map, 4.5 MB
`breach_corpus.xlsx`	Credential corpus, broken down by origin
`takeover.json`	Check of the 11 dangling records
`nuclei.jsonl`	Template scan output
`dns_control_test.txt`	The test that withdrew the §8 finding

About aihackers.inbest.cloud

This assessment service is delivered by **AIHACKERS**, an AI-driven offensive security practice, through **aihackers.inbest.cloud**, the iNBest CVP (Customer Vulnerability Program) platform. CVP runs AI-assisted, Claude-powered red-team engagements scoped to each customer's authorized surfaces and delivers reproducible, severity-ranked findings with verbatim evidence and concrete remediation.

AIHACKERS · AI-driven offensive security · CVP methodology · Powered by Claude (Anthropic)
aihackers.inbest.cloud · cybersecurity@inbest.cloud