



EXTERNAL SECURITY ASSESSMENT



External Security Assessment — docqon.com

Prepared by **Infiqon Pvt Ltd**
Generated **05 Oct 2026, 19:38 UTC** · Engine 0.1.0 · Profile **Deep**
Scope: **docqon.com**

CONFIDENTIAL — FOR THE ASSET OWNER

Executive summary

71

Security posture
· grade C

21 findings · 14 confirmed ·
0 demonstrated

4 medium

4 low

13 info

Scope

Verified assets tested (ownership proven before testing):

- docqon.com

This report presents the results of an automated, external security assessment. The highest-severity issue observed is **Content-Security-Policy header missing**. Findings marked **confirmed** were verifiably observed on the wire, and each one below shows the observation it rests on alongside step-by-step remediation. See the compliance-mapped views for SOC 2 and OWASP Top 10 alignment.

1 issue(s) previously observed on this scope have since been **re-tested and did not reproduce**. Each one, with the date it was first seen and the date the re-test cleared it, is set out under Remediation record. Issues someone marked fixed without a re-test are listed there too, separately, and are not counted in that number.

Compliance-mapped view

OWASP Top 10 (2021)

CATEGORY	FINDINGS
A02:2021 — Cryptographic Failures	2
A05:2021 — Security Misconfiguration	14

SOC 2 (Trust Services Criteria)

CRITERION	FINDINGS
CC6.1 — Logical access — restrict access to protect information assets	16
CC6.6 — Logical access — protection against threats from outside boundaries	15
CC6.7 — Restrict the transmission, movement, and removal of information	2
CC7.1 — Detect and monitor for new vulnerabilities and misconfigurations	14

Mapping indicates which controls these external findings provide evidence toward; it is not a certification of compliance.

Trust Services criteria evidenced only in part

These criteria have findings against them above, but not from every angle an external test would need. Read the counts with this table beside them.

CRITERION	WHAT IS MISSING
CC6.6 — Logical access — protection against threats from outside boundaries	Evidenced in part only. This criterion is also informed by OWASP A04:2021 Insecure Design, A10:2021 Server-Side Request Forgery (SSRF), which this assessment did not cover, so the count above rests on the remaining categories alone.
CC7.1 — Detect and monitor for new vulnerabilities and misconfigurations	Evidenced in part only. This criterion is also informed by OWASP A04:2021 Insecure Design, which this assessment did not cover, so the count above rests on the remaining categories alone.

Trust Services criteria this assessment did not evidence

Stated so an absent criterion is never read as a passing one — a table listing only what matched leaves the rest looking clean. **Inherent** means no external test reaches it, on any profile; **not this scan** means the stage that would have reached it did not run here.

CRITERION	WHY NOT
CC7.2 — Monitor system components for anomalies inherent	No part of this criterion was assessed. Externally it is evidenced only through OWASP A09:2021 Security Logging and Monitoring Failures, which this assessment did not cover — see “Categories this assessment did not cover” for why.

Categories this assessment did not cover

Stated so this report is not read as covering all ten. **Inherent** means no external test could reach it, on any profile — it is a limit of black-box testing itself. **Not this scan** means the stage that would have

covered it did not run here, so draw no conclusion either way; the “What this scan performed” table under Methodology says why.

CATEGORY	WHY NOT
A04:2021 — Insecure Design inherent	Insecure Design is a property of how controls were designed, not of what a host returns. Probing for it externally means hammering authentication and rate limits with abusive traffic, which this platform will not send at your production systems.
A09:2021 — Security Logging and Monitoring Failures inherent	Security Logging and Monitoring Failures describes what your systems record and alert on internally. Nothing observable from outside reveals it, so no external test can honestly report on it.
A10:2021 — Server-Side Request Forgery (SSRF) not this scan	Server-Side Request Forgery is confirmed by making the target call back to a server we control — out-of-band interaction testing. That stage did not run on this scan, so SSRF was not assessed.

Findings (21)

Content-Security-Policy header missing

MEDIUM

A05:2021

CWE-693

web

tool: custom:http-headers

confirmed

IMPACT

A CSP is the strongest defense-in-depth control against cross-site scripting and data-injection; its absence leaves XSS unmitigated. All 9 pages this scan read are missing it.

REMEDIATION

Define a Content-Security-Policy starting from "default-src 'self'" and tighten script/style sources; roll out with report-only first.

Apply this configuration

This host answers from behind Cloudflare, which replaces the origin's `Server` header with its own — so what runs at the origin is not visible from outside. Set this at the edge with the rule below, and at the origin too, so it survives a change of CDN.

Decide first:

- Start Report-Only, as below. It blocks nothing — the browser console lists every resource the policy would have refused, and that list is the inventory you need.
- Switch to the enforcing header (Content-Security-Policy) only when the console is quiet, having added the origins you actually use — analytics, fonts, payment widgets, your own CDN — to script-src, style-src, connect-src and img-src.
- Do not silence violations with 'unsafe-inline' or 'unsafe-eval'. Both give back most of what a CSP is for. Inline scripts are moved to files or given a per-response nonce instead.

Cloudflare — Transform Rule

```
Rules → Transform Rules → Modify Response Header → Create rule
Rule name:  Add Content-Security-Policy-Report-Only
If:         Hostname equals docqon.com
Then:       Set static
  Header name: Content-Security-Policy-Report-Only
  Value:       default-src 'self'; img-src 'self' data;; object-src 'none'; base-
uri 'self'; frame-ancestors 'self'
```

This sets the header at the edge only. Requests that reach the origin another way (a direct IP, a bypassed cache) will not carry it, so set it at the origin as well.

Any server — the header and value, which are the part that matters

```
Content-Security-Policy-Report-Only: default-src 'self'; img-src 'self' data;; object-
src 'none'; base-uri 'self'; frame-ancestors 'self'
```

Send this on every response from this host, including redirects and error pages.

Check it worked

```
curl -sI https://docqon.com | grep -i 'content-security-policy-report-only'
```

EVIDENCE

```
GET https://docqon.com/
GET https://docqon.com/integrations/
GET https://docqon.com/security/
GET https://docqon.com/pricing/
GET https://docqon.com/app/
GET https://docqon.com/app/?signup=1
GET https://docqon.com/contact/
GET https://docqon.com/privacy/
GET https://docqon.com/terms/
HTTP 200
server: cloudflare
GET https://docqon.com/

HTTP 200
age: 14999
alt-svc: h3=":443"; ma=86400
cache-control: public, max-age=0, s-maxage=86400, stale-while-revalidate=604800, must-revalidate
cf-cache-status: HIT
cf-ray: [redacted]
connection: keep-alive
content-type: text/html; charset=utf-8
date: Sun, 13 Sep 2026 12:56:41 GMT
last-modified: Tue, 25 Aug 2026 23:26:03 GMT
nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
report-to: {"group":"cf-nel","max_age":604800,"endpoints":
[{"url":"https://a.nel.cloudflare.com/report/v4?
[signed endpoint token redacted]
server: cloudflare
server-timing: cfEdge;dur=9,cfOrigin;dur=0
transfer-encoding: chunked
vary: Accept-Encoding
via: [origin server redacted]

[re-test] 'content-security-policy' absent from the response above
```

VJaF

Deprecated TLS 1.0 is enabled

MEDIUM

A02:2021

CWE-327

tls

tool: custom:tls

confirmed

IMPACT

TLS 1.0 is obsolete and vulnerable to known downgrade/crypto attacks, yet the server still negotiates it.

REMEDIATION

Disable TLS 1.0; require TLS 1.2+ (prefer 1.3).

Apply this configuration

This host answers from behind Cloudflare, which replaces the origin's `Server` header with its own — so what runs at the origin is not visible from outside. Set this at the edge with the rule below, and at the origin too, so it survives a change of CDN.

Decide first:

- Clients older than roughly 2014 cannot do TLS 1.2 — Android 4.3 and earlier, IE on unpatched Windows 7, and some payment terminals. If any of those are real users of this host, check the access logs before disabling the old versions.

Cloudflare

```
SSL/TLS → Edge Certificates → Minimum TLS Version → TLS 1.2
(per-hostname; set it for <your-domain>)
```

This governs browsers talking to Cloudflare. The Cloudflare → origin leg is set separately under SSL/TLS → Origin Server.

Any server

```
offer TLS 1.2 and TLS 1.3 only; disable TLS 1.0, TLS 1.1 and all SSL versions
```

Check it worked

```
openssl s_client -connect <your-domain>:443 -tls1_1 </dev/null # must fail
```

EVIDENCE

```
TLS handshake docqon.com:443 (SNI=docqon.com, offering only TLS 1.0)
handshake completed – the server accepted TLS 1.0
TLS handshake docqon.com:443 (SNI=docqon.com, offering only TLS 1.0)
handshake completed – negotiated TLSv1
```

Deprecated TLS 1.1 is enabled

MEDIUM

A02:2021

CWE-327

tls

tool: custom:tls

confirmed

IMPACT

TLS 1.1 is obsolete and vulnerable to known downgrade/crypto attacks, yet the server still negotiates it.

REMEDIATION

Disable TLS 1.1; require TLS 1.2+ (prefer 1.3).

Apply this configuration

This host answers from behind Cloudflare, which replaces the origin's `Server` header with its own — so what runs at the origin is not visible from outside. Set this at the edge with the rule below, and at the origin too, so it survives a change of CDN.

Decide first:

- Clients older than roughly 2014 cannot do TLS 1.2 — Android 4.3 and earlier, IE on unpatched Windows 7, and some payment terminals. If any of those are real users of this host, check the access logs before disabling the old versions.

Cloudflare

```
SSL/TLS → Edge Certificates → Minimum TLS Version → TLS 1.2
(per-hostname; set it for <your-domain>)
```

This governs browsers talking to Cloudflare. The Cloudflare → origin leg is set separately under SSL/TLS → Origin Server.

Any server

```
offer TLS 1.2 and TLS 1.3 only; disable TLS 1.0, TLS 1.1 and all SSL versions
```

Check it worked

```
openssl s_client -connect <your-domain>:443 -tls1_1 </dev/null # must fail
```

EVIDENCE

```
TLS handshake docqon.com:443 (SNI=docqon.com, offering only TLS 1.1)
handshake completed – the server accepted TLS 1.1
TLS handshake docqon.com:443 (SNI=docqon.com, offering only TLS 1.1)
handshake completed – negotiated TLSv1.1
```

HSTS header missing

MEDIUM

A05:2021

CWE-693

web

tool: custom:http-headers

confirmed

IMPACT

Without HTTP Strict Transport Security, a network attacker can strip TLS and downgrade visitors to plaintext HTTP on the first request. All 9 pages this scan read are missing it.

REMEDIATION

Add 'Strict-Transport-Security: max-age=31536000; includeSubDomains' (consider 'preload') on all HTTPS responses.

Apply this configuration

This host answers from behind Cloudflare, which replaces the origin's `Server` header with its own — so what runs at the origin is not visible from outside. Set this at the edge with the rule below, and at the origin too, so it survives a change of CDN.

Decide first:

- includeSubDomains commits every subdomain of this host to HTTPS, including ones you have not moved yet — an internal tool still on http:// will stop loading. Confirm they all serve HTTPS, or send the header without that directive until they do.
- Add '; preload' only if you intend to submit the domain to the browser preload list. Removal from that list takes months, so it is not a decision to make by pasting.

Cloudflare — Transform Rule

```
Rules → Transform Rules → Modify Response Header → Create rule
Rule name:  Add Strict-Transport-Security
If:         Hostname equals docqon.com
Then:       Set static
Header name: Strict-Transport-Security
Value:      max-age=31536000; includeSubDomains
```

Cloudflare also has a purpose-built control for this one: SSL/TLS → Edge Certificates → HTTP Strict Transport Security (HSTS) → Enable. Use that instead of a Transform Rule if you are only setting HSTS. This sets the header at the edge only. Requests that reach the origin another way (a direct IP, a bypassed cache) will not carry it, so set it at the origin as well.

Any server — the header and value, which are the part that matters

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
```

Send this on every response from this host, including redirects and error pages.

Check it worked

```
curl -sI https://docqon.com | grep -i 'strict-transport-security'
```

EVIDENCE

```
GET https://docqon.com/
GET https://docqon.com/integrations/
GET https://docqon.com/security/
GET https://docqon.com/pricing/
GET https://docqon.com/app/
GET https://docqon.com/app/?signup=1
GET https://docqon.com/contact/
GET https://docqon.com/privacy/
GET https://docqon.com/terms/
HTTP 200
server: cloudflare
GET https://docqon.com/

HTTP 200
age: 14999
alt-svc: h3=":443"; ma=86400
cache-control: public, max-age=0, s-maxage=86400, stale-while-revalidate=604800, must-revalidate
cf-cache-status: HIT
cf-ray: [redacted]
connection: keep-alive
content-type: text/html; charset=utf-8
date: Sun, 13 Sep 2026 12:56:41 GMT
last-modified: Tue, 25 Aug 2026 23:26:03 GMT
nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
report-to: {"group":"cf-nel","max_age":604800,"endpoints":
[{"url":"https://a.nel.cloudflare.com/report/v4?
[signed endpoint token redacted]
server: cloudflare
server-timing: cfEdge;dur=9,cfOrigin;dur=0
transfer-encoding: chunked
vary: Accept-Encoding
via: [origin server redacted]

[re-test] 'strict-transport-security' absent from the response above
```

VJaF

Clickjacking protection missing

LOW

A05:2021

CWE-693

web

tool: custom:http-headers

confirmed

IMPACT

Without X-Frame-Options or a CSP frame-ancestors directive, the site can be framed by a third party for clickjacking. All 9 pages this scan read are missing it.

REMEDIATION

Send 'X-Frame-Options: DENY' (or 'SAMEORIGIN'), or set CSP "frame-ancestors 'self'".

Apply this configuration

This host answers from behind Cloudflare, which replaces the origin's `Server` header with its own — so what runs at the origin is not visible from outside. Set this at the edge with the rule below, and at the origin too, so it survives a change of CDN.

Decide first:

- SAMEORIGIN is the safe default: your own pages can still frame each other, nobody else can. Use DENY instead if nothing on this site is ever framed, and neither if you deliberately embed this site in a partner's page — in that case use Content-Security-Policy: frame-ancestors with their origin listed.

Cloudflare — Transform Rule

```
Rules → Transform Rules → Modify Response Header → Create rule
Rule name: Add X-Frame-Options
If:        Hostname equals docqon.com
Then:      Set static
Header name: X-Frame-Options
Value:     SAMEORIGIN
```

This sets the header at the edge only. Requests that reach the origin another way (a direct IP, a bypassed cache) will not carry it, so set it at the origin as well.

Any server — the header and value, which are the part that matters

```
X-Frame-Options: SAMEORIGIN
```

Send this on every response from this host, including redirects and error pages.

Check it worked

```
curl -sI https://docqon.com | grep -i 'x-frame-options'
```

EVIDENCE

```
GET https://docqon.com/
GET https://docqon.com/integrations/
GET https://docqon.com/security/
GET https://docqon.com/pricing/
```

```
GET https://docqon.com/app/
GET https://docqon.com/app/?signup=1
GET https://docqon.com/contact/
GET https://docqon.com/privacy/
GET https://docqon.com/terms/
HTTP 200
server: cloudflare
GET https://docqon.com/

HTTP 200
age: 14999
alt-svc: h3=":443"; ma=86400
cache-control: public, max-age=0, s-maxage=86400, stale-while-revalidate=604800, must-revalidate
cf-cache-status: HIT
cf-ray: [redacted]
connection: keep-alive
content-type: text/html; charset=utf-8
date: Sun, 13 Sep 2026 12:56:41 GMT
last-modified: Tue, 25 Aug 2026 23:26:03 GMT
nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
report-to: {"group":"cf-nel","max_age":604800,"endpoints":
[{"url":"https://a.nel.cloudflare.com/report/v4?
[signed endpoint token redacted]
server: cloudflare
server-timing: cfEdge;dur=9,cfOrigin;dur=0
transfer-encoding: chunked
vary: Accept-Encoding
via: [origin server redacted]

[re-test] 'x-frame-options' absent from the response above
```

VJaF

DMARC policy is monitor-only (p=none)**LOW**

A05:2021

CWE-16

dns

tool: custom:dns-email

confirmed

IMPACT

DMARC is present but set to 'p=none', so spoofed mail is not actually rejected.

REMEDIATION

After reviewing reports, tighten to 'p=quarantine' then 'p=reject'.

Apply this configuration

This is a DNS record, not web-server configuration — publish it wherever this domain's DNS is hosted. It takes effect once the old answer's TTL has passed.

Decide first:

- pct=25 applies the policy to a quarter of failing mail, so a sender you forgot shows up in the reports before it costs you all of that mail. Raise it to 100, then move p to reject.
- Before raising it, confirm every legitimate sender passes SPF or DKIM — the aggregate reports at the rua address list them.

Zone file

```
_dmarc.docqon.com. IN TXT "v=DMARC1; p=quarantine; pct=25; rua=mailto:dmarc@docqon.com"
```

Or, in a DNS provider's form (Cloudflare, Route 53, GoDaddy...)

```
Type:  TXT
Name:  _dmarc.docqon.com
Value: v=DMARC1; p=quarantine; pct=25; rua=mailto:dmarc@docqon.com
TTL:   3600
```

Some panels want the name relative to the domain — where this says _dmarc.docqon.com, they may expect _dmarc.

Then, when a fortnight of reports is clean

```
_dmarc.docqon.com. IN TXT "v=DMARC1; p=reject; rua=mailto:dmarc@docqon.com"
```

Check it worked

```
dig +short TXT _dmarc.docqon.com
```

EVIDENCE

```
dig +short TXT _dmarc.docqon.com
_dmarc.docqon.com. IN TXT "v=DMARC1; p=none; rua=mailto:dmarc@[redacted]
[redacted]; ruf=mailto:dmarc@[redacted]; rf=afrf; pct=100"
dig +noall +answer TXT _dmarc.docqon.com

v=DMARC1; p=none; rua=mailto:dmarc@[redacted]; ruf=mailto:dmarc@[redacted]
[redacted]; rf=afrf; pct=100
```

```
[re-test] still contains 'p=none'
```

Weak Cipher Suites Detection

LOW

A05:2021

web

tool: nuclei:weak-cipher-suites

unconfirmed

IMPACT

A weak cipher is defined as an encryption/decryption algorithm that uses a key of insufficient length. Using an insufficient length for a key in an encryption/decryption algorithm opens up the possibility (or probability) that the encryption scheme could be broken.

REMEDIATION

See vendor advisory for the referenced CVE.

EVIDENCE

```
matched-at: docqon.com:443
template: weak-cipher-suites
```

X-Content-Type-Options header missing

LOW

A05:2021

CWE-693

web

tool: custom:http-headers

confirmed

IMPACT

Browsers may MIME-sniff responses, enabling some content-type confusion and XSS vectors. All 9 pages this scan read are missing it.

REMEDIATION

Send 'X-Content-Type-Options: nosniff' on all responses.

Apply this configuration

This host answers from behind Cloudflare, which replaces the origin's `Server` header with its own — so what runs at the origin is not visible from outside. Set this at the edge with the rule below, and at the origin too, so it survives a change of CDN.

Cloudflare — Transform Rule

```
Rules → Transform Rules → Modify Response Header → Create rule
Rule name: Add X-Content-Type-Options
If:        Hostname equals docqon.com
Then:      Set static
Header name: X-Content-Type-Options
Value:     nosniff
```

This sets the header at the edge only. Requests that reach the origin another way (a direct IP, a bypassed cache) will not carry it, so set it at the origin as well.

Any server — the header and value, which are the part that matters

```
X-Content-Type-Options: nosniff
```

Send this on every response from this host, including redirects and error pages.

Check it worked

```
curl -sI https://docqon.com | grep -i 'x-content-type-options'
```

EVIDENCE

```
GET https://docqon.com/
GET https://docqon.com/integrations/
GET https://docqon.com/security/
GET https://docqon.com/pricing/
GET https://docqon.com/app/
GET https://docqon.com/app/?signup=1
GET https://docqon.com/contact/
GET https://docqon.com/privacy/
GET https://docqon.com/terms/
HTTP 200
server: cloudflare
```

```
GET https://docqon.com/
```

```
HTTP 200
age: 14999
alt-svc: h3=":443"; ma=86400
cache-control: public, max-age=0, s-maxage=86400, stale-while-revalidate=604800, must-revalidate
cf-cache-status: HIT
cf-ray: [redacted]
connection: keep-alive
content-type: text/html; charset=utf-8
date: Sun, 13 Sep 2026 12:56:41 GMT
last-modified: Tue, 25 Aug 2026 23:26:03 GMT
nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
report-to: {"group":"cf-nel","max_age":604800,"endpoints":
[{"url":"https://a.nel.cloudflare.com/report/v4?
[signed endpoint token redacted]
server: cloudflare
server-timing: cfEdge;dur=9,cfOrigin;dur=0
transfer-encoding: chunked
vary: Accept-Encoding
via: [origin server redacted]
```

VJaF

[re-test] 'x-content-type-options' absent from the response above

DNS WAF Detection

INFO

CWE-200

web

tool: nuclei:dns-waf-detect

unconfirmed

IMPACT

A DNS WAF was detected.

REMEDIATION

See vendor advisory for the referenced CVE.

EVIDENCE

```
matched-at: docqon.com
template: dns-waf-detect
```

DNSSEC not enabled

INFO

A05:2021

CWE-350

dns

tool: custom:dns-email

confirmed

IMPACT

Without DNSSEC, DNS answers for this domain are not cryptographically signed and can be spoofed by an on-path attacker.

REMEDIATION

Enable DNSSEC at your DNS provider and publish DS records at the registrar.

Apply this configuration

DNSSEC is switched on at your DNS host, which then generates a DS record that only it can produce — so there is no value here to paste. What follows is the sequence, and the check that it worked.

Decide first:

- Steps 1 and 3 must both happen, in that order. A DS record published at the registrar while the DNS host is not signing — or left behind after you move DNS providers — makes the domain fail to resolve for every validating resolver. That is a full outage, not a degraded one.

The sequence

1. DNS host (Cloudflare, Route 53, your registrar's DNS): enable DNSSEC for docqon.com.
2. It shows you a DS record — key tag, algorithm, digest type, digest.
3. Registrar (where the domain was bought): paste that DS record into the domain's DNSSEC section. This is often a different company from step 1.
4. Wait for the parent zone to publish it, then run the check below.

The DS record's digest is generated from your zone's own key. Any value written here would be fiction, which is why there is none.

Check it worked

```
dig +dnssec docqon.com | grep -c RRSIG    # 0 means still unsigned
```

EVIDENCE

```
dig +short DNSKEY docqon.com
;; no records returned
dig +noall +answer DNSKEY docqon.com

;; no records returned

[re-test] no record of this type at this name
```

Detect Sentry Instance

INFO

A10:2021

web

tool: nuclei:detect-sentry

unconfirmed

REMEDIATION

See vendor advisory for the referenced CVE.

EVIDENCE

```
matched-at: https://docqon.com/app/assets/index-[hash].js
template: detect-sentry
```

FingerprintHub Technology Fingerprint

INFO

CWE-200

web

tool: nuclei:fingerprinthub-web-fingerprints

unconfirmed

IMPACT

FingerprintHub Technology Fingerprint tests run in nuclei.

REMEDIATION

See vendor advisory for the referenced CVE.

EVIDENCE

```
matched-at: https://docqon.com/app/assets/index-[hash].js
template: fingerprinthub-web-fingerprints
```

HTTP Missing Security Headers

INFO

A05:2021

CWE-693

web

tool: nuclei:http-missing-security-headers

unconfirmed

IMPACT

This template searches for missing HTTP security headers. The impact of these missing headers can vary.

REMEDIATION

See vendor advisory for the referenced CVE.

EVIDENCE

```
matched-at: https://docqon.com/terms/
template: http-missing-security-headers
```

Jan AI - Detect

INFO

web

tool: nuclei:jan-detect

unconfirmed

IMPACT

Detects the Jan AI local server, an open-source ChatGPT alternative that runs models locally and exposes a Cortex-powered, OpenAI-compatible API (default port 1337).

REMEDIATION

See vendor advisory for the referenced CVE.

EVIDENCE

```
matched-at: https://docqon.com/healthz  
template: jan-detect
```

No CAA record

INFO

A05:2021

CWE-295

dns

tool: custom:dns-email

confirmed

IMPACT

A CAA record restricts which CAs may issue certificates for this domain, reducing mis-issuance risk.

REMEDIATION

Add a CAA record naming your CA(s), e.g. '0 issue "letsencrypt.org"'.

Apply this configuration

This is a DNS record, not web-server configuration — publish it wherever this domain's DNS is hosted. It takes effect once the old answer's TTL has passed.

Decide first:

- The value has to name the CA that actually issues your certificates. Naming the wrong one does not fail loudly — it fails at your next renewal, when the CA reads the record and refuses to issue.
- If a CDN or host manages certificates for you (Cloudflare, Vercel, AWS, Netlify), it may use several CAs and change between them. Publish the identifiers that provider documents, or leave CAA unset — an absent CAA record is a low-severity finding, a wrong one is a failed renewal.

Zone file

```
docqon.com. IN CAA 0 issue "letsencrypt.org"
```

Or, in a DNS provider's form (Cloudflare, Route 53, GoDaddy...)

```
Type:  CAA
Name:  docqon.com
Value: 0 issue "letsencrypt.org"
TTL:   3600
```

Some panels want the name relative to the domain — where this says docqon.com, they may expect @.

The identifiers of the common certificate authorities

Let's Encrypt	letsencrypt.org
Google Trust Services	pki.goog
DigiCert	digicert.com
Sectigo	sectigo.com
Amazon (ACM)	amazon.com

```
# add a second record to be told about refused requests:
docqon.com. IN CAA 0 iodef "mailto:security@docqon.com"
```

Check it worked

```
dig +short CAA docqon.com
```

EVIDENCE

```
dig +short CAA docqon.com
;; no records returned
dig +noall +answer CAA docqon.com

;; no records returned

[re-test] no record of this type at this name
```

Port 443/tcp (https) is reachable from the internet

INFO

A05:2021

CWE-668

network

tool: custom:ports

confirmed

IMPACT

This is a normal public web or mail service. It is listed so that a change to your exposed surface is visible: if this port closes, or a new one opens beside it, the next scan will say so. Note: Cloudflare answers for this hostname, so this port was observed on the Cloudflare edge rather than on your own server. Cloudflare opens a fixed set of ports on its shared addresses for every site it serves; you cannot close them, and their being open does not mean the corresponding port is open on your origin. What this does tell you is which entry points Cloudflare will accept traffic on for your site.

REMEDIATION

Because this port belongs to the Cloudflare edge and not to your server, there is nothing to close here. If you want traffic restricted to fewer entry points, that is a Cloudflare configuration change. Separately, confirm your origin only accepts connections from Cloudflare so the origin cannot be reached directly, bypassing it.

EVIDENCE

```
TCP connect docqon.com:443
connection established to docqon.com:443 (https) – answered by the Cloudflare edge, not
the origin
TCP connect docqon.com:443

connection established to docqon.com:443
```

Port 80/tcp (http) is reachable from the internet

INFO

A05:2021

CWE-668

network

tool: custom:ports

confirmed

IMPACT

This is a normal public web or mail service. It is listed so that a change to your exposed surface is visible: if this port closes, or a new one opens beside it, the next scan will say so. Note: Cloudflare answers for this hostname, so this port was observed on the Cloudflare edge rather than on your own server. Cloudflare opens a fixed set of ports on its shared addresses for every site it serves; you cannot close them, and their being open does not mean the corresponding port is open on your origin. What this does tell you is which entry points Cloudflare will accept traffic on for your site.

REMEDIATION

Because this port belongs to the Cloudflare edge and not to your server, there is nothing to close here. If you want traffic restricted to fewer entry points, that is a Cloudflare configuration change. Separately, confirm your origin only accepts connections from Cloudflare so the origin cannot be reached directly, bypassing it.

EVIDENCE

```
TCP connect docqon.com:80
connection established to docqon.com:80 (http) – answered by the Cloudflare edge, not
the origin
TCP connect docqon.com:80

connection established to docqon.com:80
```

Port 8080/tcp (http-alt) is reachable from the internet

INFO

A05:2021

CWE-668

network

tool: custom:ports

confirmed

IMPACT

A web service is answering on a non-standard port. That is often deliberate, but it is also where staging builds, admin consoles and debug listeners end up exposed by accident, so it is worth confirming that this one is meant to be public. Note: Cloudflare answers for this hostname, so this port was observed on the Cloudflare edge rather than on your own server. Cloudflare opens a fixed set of ports on its shared addresses for every site it serves; you cannot close them, and their being open does not mean the corresponding port is open on your origin. What this does tell you is which entry points Cloudflare will accept traffic on for your site.

REMEDIATION

Because this port belongs to the Cloudflare edge and not to your server, there is nothing to close here. If you want traffic restricted to fewer entry points, that is a Cloudflare configuration change. Separately, confirm your origin only accepts connections from Cloudflare so the origin cannot be reached directly, bypassing it.

EVIDENCE

```
TCP connect docqon.com:8080
connection established to docqon.com:8080 (http-alt) – answered by the Cloudflare edge,
not the origin
TCP connect docqon.com:8080

connection established to docqon.com:8080
```

Port 8443/tcp (https-alt) is reachable from the internet

INFO

A05:2021

CWE-668

network

tool: custom:ports

confirmed

IMPACT

A web service is answering on a non-standard port. That is often deliberate, but it is also where staging builds, admin consoles and debug listeners end up exposed by accident, so it is worth confirming that this one is meant to be public. Note: Cloudflare answers for this hostname, so this port was observed on the Cloudflare edge rather than on your own server. Cloudflare opens a fixed set of ports on its shared addresses for every site it serves; you cannot close them, and their being open does not mean the corresponding port is open on your origin. What this does tell you is which entry points Cloudflare will accept traffic on for your site.

REMEDIATION

Because this port belongs to the Cloudflare edge and not to your server, there is nothing to close here. If you want traffic restricted to fewer entry points, that is a Cloudflare configuration change. Separately, confirm your origin only accepts connections from Cloudflare so the origin cannot be reached directly, bypassing it.

EVIDENCE

```
TCP connect docqon.com:8443
connection established to docqon.com:8443 (https-alt) – answered by the Cloudflare
edge, not the origin
TCP connect docqon.com:8443

connection established to docqon.com:8443
```

Referrer-Policy header missing

INFO

A05:2021

CWE-693

web

tool: custom:http-headers

confirmed

IMPACT

The full URL (which may carry sensitive path/query data) can leak to third parties via the Referer header. All 9 pages this scan read are missing it.

REMEDIATION

Send 'Referrer-Policy: strict-origin-when-cross-origin' or stricter.

Apply this configuration

This host answers from behind Cloudflare, which replaces the origin's `Server` header with its own — so what runs at the origin is not visible from outside. Set this at the edge with the rule below, and at the origin too, so it survives a change of CDN.

Cloudflare — Transform Rule

```
Rules → Transform Rules → Modify Response Header → Create rule
Rule name: Add Referrer-Policy
If:        Hostname equals docqon.com
Then:      Set static
Header name: Referrer-Policy
Value:      strict-origin-when-cross-origin
```

This sets the header at the edge only. Requests that reach the origin another way (a direct IP, a bypassed cache) will not carry it, so set it at the origin as well.

Any server — the header and value, which are the part that matters

```
Referrer-Policy: strict-origin-when-cross-origin
```

Send this on every response from this host, including redirects and error pages.

Check it worked

```
curl -sI https://docqon.com | grep -i 'referrer-policy'
```

EVIDENCE

```
GET https://docqon.com/
GET https://docqon.com/integrations/
GET https://docqon.com/security/
GET https://docqon.com/pricing/
GET https://docqon.com/app/
GET https://docqon.com/app/?signup=1
GET https://docqon.com/contact/
GET https://docqon.com/privacy/
GET https://docqon.com/terms/
HTTP 200
server: cloudflare
```

```
GET https://docqon.com/
```

```
HTTP 200
age: 14999
alt-svc: h3=":443"; ma=86400
cache-control: public, max-age=0, s-maxage=86400, stale-while-revalidate=604800, must-revalidate
cf-cache-status: HIT
cf-ray: [redacted]
connection: keep-alive
content-type: text/html; charset=utf-8
date: Sun, 13 Sep 2026 12:56:41 GMT
last-modified: Tue, 25 Aug 2026 23:26:03 GMT
nel: {"report_to":"cf-nel","success_fraction":0.0,"max_age":604800}
report-to: {"group":"cf-nel","max_age":604800,"endpoints":
[{"url":"https://a.nel.cloudflare.com/report/v4?
[signed endpoint token redacted]
server: cloudflare
server-timing: cfEdge;dur=9,cfOrigin;dur=0
transfer-encoding: chunked
vary: Accept-Encoding
via: [origin server redacted]
```

VJaf

[re-test] 'referrer-policy' absent from the response above

Wappalyzer Technology Detection

INFO

web

tool: nuclei:tech-detect

unconfirmed

REMEDIATION

See vendor advisory for the referenced CVE.

EVIDENCE

```
matched-at: https://docqon.com/api/config
template: tech-detect
```

Remediation record

Issues that were observed on this scope and have since stopped appearing, with the date each was first seen and what — if anything — established that it is gone. Every date below is the date of the scan that observed it, so this section reads the same whenever the document is regenerated.

Only the first group is evidence. A fix is proven when a later scan ran the very check that found the issue and did not find it again. An issue somebody marked fixed is a statement by the asset owner; an issue that merely stopped appearing may have been fixed, or the check may not have run that time. These are kept apart on purpose and are never added together into a single “fixed” count.

Verified fixed — re-tested and did not reproduce (1)

Each of these was found again by the same check on a later date, and was not there. This is the closure evidence an auditor can rely on.

WAF Detection

INFO

web

Known-vulnerability template scan

RE-TESTED CLEAN

First observed **12 Sep 2026** · last observed **13 Sep 2026** · re-tested clean **13 Sep 2026** · observed on 3 scans

EVIDENCE OF CLOSURE

First observed 12 Sep 2026. On 13 Sep 2026 the known-vulnerability template scan ran again against the same asset and did not reproduce it. That re-test is the evidence this issue is closed; the stage recorded: community template scan of 14 URL(s) on <https://docqon.com/> — the safe set in 15040s, of a 28025s budget. Tags cve,tech,misconfig,misconfiguration,exposure,default-login,takeover; excluded intrusive,dos,fuzz,brute-force. 214 match(es). That is every same-host URL the crawl found. 143 request(s) were paced by the per-asset rate limit

Re-test: Deep scan [scan id redacted], engine 0.1.0.

Reported fixed — not yet re-tested (3)

Marked fixed by your team. Nothing has re-tested them, so they are recorded here as the asset owner's statement rather than as a result. Run a scan covering the asset and each one moves into the group above — or reopens.

Deprecated TLS 1.1 is enabled

MEDIUM

tls

A02:2021

TLS and certificate check

OWNER'S CLAIM

First observed **29 Aug 2026** · last observed **13 Sep 2026** · not re-tested

WHAT IS ESTABLISHED

First observed 29 Aug 2026, last observed 13 Sep 2026. Marked fixed by your team. No scan has covered this asset since, so nothing has re-tested it. This is a statement by the asset owner, not a verified result.

Deprecated TLS 1.0 is enabled

MEDIUM

tls

A02:2021

TLS and certificate check

OWNER'S CLAIM

First observed **29 Aug 2026** · last observed **13 Sep 2026** · not re-tested

WHAT IS ESTABLISHED

First observed 29 Aug 2026, last observed 13 Sep 2026. Marked fixed by your team. No scan has covered this asset since, so nothing has re-tested it. This is a statement by the asset owner, not a verified result.

Weak Cipher Suites Detection

LOW

web

A05:2021

Known-vulnerability template scan

OWNER'S CLAIMFirst observed **12 Sep 2026** · last observed **13 Sep 2026** · not re-tested

WHAT IS ESTABLISHED

First observed 12 Sep 2026, last observed 13 Sep 2026. Marked fixed by your team. No scan has covered this asset since, so nothing has re-tested it. This is a statement by the asset owner, not a verified result.

Stopped appearing — nothing has verified it (1)

These are listed for completeness and are **not** claimed as fixed. An issue leaves a scan for three reasons — it was fixed, the check did not run, or the host did not answer — and on these we cannot tell which. Absence is not a result.

WAF blocked 1 of 8 attack signatures

INFO

web

A05:2021

WAF effectiveness test

NOT VERIFIEDFirst observed **11 Sep 2026** · last observed **11 Sep 2026** · not re-tested

WHAT IS ESTABLISHED

First observed 11 Sep 2026, last observed 11 Sep 2026. It has not appeared since, but the check that finds it did not complete on any scan since (aggressive scanning was not acknowledged, so no attack signature was sent and the WAF was not tested.). An issue can vanish from a scan because it was fixed or because nothing looked, and this one has not been looked for.

Assessment summary (attestation)

This document certifies that Infiqon Pvt Ltd, via its SecQon platform, performed an automated external (deep) security assessment of the following internet-facing asset(s): docqon.com, on 05 October 2026. The assessment was conducted against assets whose ownership was verified prior to testing, and additionally attempted bounded, non-destructive proof-of-exploit techniques within an intrusiveness scope the customer acknowledged for this scan. Where an exploit was reproduced, the technique used is recorded in that finding's evidence; the stages performed and their outcomes are listed under Methodology. No testing destroyed data, denied service, established persistence, moved laterally, or extracted data in bulk. 21 finding(s) were identified and are detailed in this report with supporting evidence and remediation guidance. This summary reflects the security posture observed at the time of testing and is scoped to external, black-box, unauthenticated assessment.

Infiqon Pvt Ltd further certifies that 1 issue(s) previously observed on this scope were re-tested by the same check on the dates recorded under Remediation record and did not reproduce. Each such statement rests on a completed scan whose stage record shows the check ran; no issue is certified as fixed on the strength of its absence, nor on the strength of anyone having marked it fixed.

Methodology & scope

Testing was external and black-box, and unauthenticated. All targets had their ownership verified before any traffic was sent (domains by DNS TXT record, hosted file or meta tag; IP addresses by hosted token), and every outbound request was checked against a verified-asset allowlist at the egress boundary. On this **Deep-profile** assessment, testing went beyond observation to **bounded, non-destructive exploitation** — the smallest proof-of-exploit that demonstrates an issue — within an intrusiveness scope acknowledged for this scan; the specific technique used for each finding appears in that finding's evidence. Even so, no testing destroyed data, denied service, established persistence, moved laterally, or extracted data in bulk. Findings are grounded in tool evidence captured during the assessment.

What this scan performed

Every active stage, and what it actually did — including the ones that had nothing to test. A stage that ran and found nothing is a result, not a gap, and it is published here so this report is never read as claiming work it did not do.

STAGE	OUTCOME
Public access-control check	Ran — requested a control path that cannot exist, to prove this server's 404 really is a 404, then the 8 curated paths that leak source control, configuration and backups (<code>/.git/config</code> , <code>/.git/HEAD</code> , <code>/.env</code> , <code>/.svn/entries</code> , <code>/.DS_Store</code> , <code>/config.json</code> , <code>/wp-config.php.bak</code> , <code>/.aws/credentials</code>), then the homepage and 2 asset directories (<code>/assets/</code> , <code>/uploads/</code>) for a server-generated file index; 14 pages — the homepage and the URLs the crawler discovered — were each asked to accept a credentialed cross-origin request from an arbitrary website, because that policy is set per route. A 200 counted only where the body looked like the file it claimed to be. Every request was an anonymous GET, and the path list is fixed — a sensitive file at an address nobody guesses was not looked for.
Active vulnerability scan (aggressive)	Not run — aggressive active scanning was not acknowledged for this scan, so no attack payload was sent. The path-traversal, template-injection, command-injection, SSRF and related tests were NOT run.

STAGE	OUTCOME
API schema and endpoint check	Not run — looked for an API contract under https://docqon.com/ and found none: no OpenAPI or Swagger document answered at the 5 conventional locations (/openapi.json, /swagger.json, /v3/api-docs, /api-docs, /swagger/v1/swagger.json) and no GraphQL endpoint answered a query at /graphql, so there was nothing to enumerate operations from and nothing schema-aware was tested (6 request(s) sent). This is a fixed list of conventional locations, not a search: endpoint names are never guessed, so an API documented at an address nobody publishes was not found here. Documents in YAML rather than JSON are not read.
Authentication-configuration check	Ran — read the homepage (https://docqon.com/) for a Set-Cookie carrying no SameSite attribute, and read 4 page(s) served without signing in (https://docqon.com/ , https://docqon.com/login , https://docqon.com/signin , https://docqon.com/account/login) for a form whose action is a plaintext http:// address. No credential was submitted, no login was attempted and no password or lockout policy was probed, so nothing here speaks to how your authentication behaves once someone signs in.
DNS and email-record check	Ran — queried public DNS for docqon.com and, where the record type is inherited, its parent domains: SPF and DMARC (TXT), CAA, and DNSKEY for DNSSEC. Every absence was confirmed against a control name known to publish that record type, so a resolver that cannot serve it is never mistaken for a domain that does not publish it; 3 problem(s) reported. DKIM is not covered — a domain's selector names cannot be listed from outside it.
Form input reflection probe	Not run — 0 form(s) submit with POST and none could be shown to be a search rather than an action, so nothing was submitted. This engine only presses a form it can positively identify as a query — a form that records, sends, pays or signs in is left alone, because submitting one would change your data. Input reflected through those forms was therefore NOT tested.

STAGE	OUTCOME
Form security check	Ran — read the markup of 9 page(s) already fetched and found 0 form(s), 0 of which submit with POST; checked each for a CSRF token field, for an action pointing at another origin, and for a file upload; 0 problem(s) reported. No form was submitted and no request was sent by this check — it reads responses the scan had already collected. A CSRF finding here reports missing markup, not a forged submission the server accepted: proving that would mean performing the state change the form exists to make.
Injection precondition check	Ran — sent one inert marker through a query parameter on 14 page(s) — https://docqon.com/ and 13 URL(s) the crawler discovered — and checked whether it came back inside the HTML with its angle brackets intact, then read every response body for a database engine's own error text; 0 problem(s) reported. No quote, metacharacter or payload was sent and nothing was changed, so this establishes the preconditions for injection rather than whether an attack would succeed. The crawl is capped at 100 pages per scan, so a page it did not reach was not probed; anything behind a sign-in was not visited.
Endpoint discovery (crawl)	Ran — crawled https://docqon.com/ to depth 3 in 143s — 14 same-host URL(s) found, 1 with query parameters to test. It ran out of links to follow before either limit, so this is the site's same-host surface at this depth. Pages were rendered in a headless browser, so routes and API calls a single-page application only makes at runtime were followed, and endpoints were parsed out of its JavaScript. The crawl was unauthenticated, so only the public surface was walked — pages behind a login were not discovered. Store credentials for this asset and acknowledge authenticated testing to include them.
Mass-assignment test	Not run — aggressive scanning was not acknowledged, so no authorization test was run
Known-vulnerability template scan	Ran — community template scan of 14 URL(s) on https://docqon.com/ — the safe set in 15040s, of a 28025s budget. Tags cve,tech,misconfig,misconfiguration,exposure,default-login,takeover; excluded intrusive,dos,fuzz,brute-force. 214 match(es). That is every same-host URL the crawl found. 143 request(s) were paced by the per-asset rate limit
Object-level authorization (IDOR/BOLA)	Not run — aggressive scanning was not acknowledged, so no authorization test was run

STAGE	OUTCOME
Exposed-service (port) check	Ran — 36 of 36 curated TCP ports probed (21, 22, 23, 25, 80, 110, 143, 443, 445, 1433, 1521, 2375, 3306, 3389, 5432, 5601, 5900, 6379, 8080, 8443, 9200, 11211, 27017, 2376, 2379, 6443, 10250, 5984, 8086, 9042, 9300, 5672, 15672, 8500, 9000, 3000); 4 reachable. This is a targeted list of commonly exposed services, not a sweep of all 65,535 ports, and UDP was not examined — a service on any other port would not appear in this report.
Reflected injection proof-of-exploit	Ran — injected inert markup into the homepage query string; the page escaped or dropped it — correct behaviour, no reflected injection
Security-header check	Ran — read the response headers of 9 HTML page(s) — https://docqon.com/ and 8 URL(s) the crawler discovered — looking for 5 browser-hardening headers (Strict-Transport-Security, Content-Security-Policy, X-Content-Type-Options, X-Frame-Options, Referrer-Policy) and, on every Set-Cookie, the Secure and HttpOnly flags; 5 problem(s) reported. Where a header was present its value was checked for the cases that make it ineffective — an HSTS max-age under six months, a Content-Security-Policy that does not restrict scripts or permits inline ones, a clickjacking or nosniff value browsers ignore, and a referrer policy that sends full URLs off-site. That is a fixed list of unambiguous faults, not a review of whether the policy is well designed for this application. 5 discovered URL(s) were not HTML and were not judged on these headers, which govern how a browser renders a document; their cookies were still read. Anything behind a sign-in was not visited.
Authenticated-session validity	Not run — no session is stored for this asset, so there was nothing to validate
SQL injection testing	Ran — boolean-based SQL injection probed on 1 of the 1 parameterized URL(s) discovered, detection only (no data read) — 0 confirmed
Out-of-band SSRF confirmation	Not run — no URL parameter looked capable of fetching a URL

STAGE	OUTCOME
Supply-chain integrity check	Ran — read the markup of 9 page(s) on this host for <code><script></code> elements loading code from another host with no Subresource Integrity hash, then requested a control path that cannot exist followed by the 6 curated build-metadata paths (<code>/.github/workflows/</code> , <code>/.gitlab-ci.yml</code> , <code>/Jenkinsfile</code> , <code>/.travis.yml</code> , <code>/docker-compose.yml</code> , <code>/package-lock.json</code>); a 200 counted only where the body really was that file. Only these 9 page(s) and this fixed list of paths were examined, and reading which third-party code a page loads is not an inventory of the dependencies inside your application.
Technology-banner check	Ran — read the Server, X-Powered-By, X-AspNet-Version, X-Generator response headers of 14 page(s) starting at <code>https://docqon.com/</code> : 1 were present and 0 named a product together with a version number. Only these headers were examined — a version disclosed in a page body, an error page or a JavaScript bundle was not looked for.
TLS and certificate check	Ran — completed a TLS handshake with <code>docqon.com:443</code> and read the certificate's chain of trust, the hostname it covers and its expiry date, then offered TLS 1.0 and TLS 1.1 separately to see whether either is still accepted; 2 problem(s) reported. Only the certificate presented on port 443 was examined — certificates on other services were not.
WAF effectiveness test	Not run — aggressive scanning was not acknowledged, so no attack signature was sent and the WAF was not tested.

How a finding becomes "confirmed." A finding is marked **confirmed** only when a reproduction transcript backs it, and that transcript is printed under Evidence. For almost every check that means the candidate was re-tested against the live asset after the initial scan — the same request, handshake, DNS query or connection made again through the same scope guard and rate limiter — and the re-test reproduced it. A small number of proofs cannot be re-issued by repeating a request: an out-of-band callback, for instance, is your own server contacting a collector we run, and the record of that contact is itself the reproduction. A finding observed once but not reproduced is likely; one we could not re-test stays unconfirmed. An inability to re-test never counts as evidence that a finding is gone.

"Confirmed" and "demonstrated" are different claims. **Confirmed** means we went back and observed the same thing again — the header is still absent, the protocol version is still accepted, the port still answers. That is a fact about your configuration, checked twice. **Demonstrated** means a bounded proof-of-exploit changed how the application behaved: markup we injected was rendered, a database query answered differently to two benign predicates measured against a stable control, or your server called a collector we run. Only the second is exploitation, and a report showing **0** demonstrated finding(s) is saying exactly that and nothing more. Most assessments of a well-maintained host show none, which is the result you want rather than a gap in the testing.

Infiqon Pvt Ltd · SecQon automated external VAPT · This report is confidential and intended for the asset owner. Posture score is a heuristic derived from finding severities and is not a guarantee of security.