Security report for

harbourline.example

Executive summary + details

C

Overall grade

Weighted average across 4 security checks — the latest result of each type.

Produced 17 October 2026, 19:36 UTC. Every result in this report is the most recent completed run of that check. A description of each check, and how to read its grade, is at the end.

Contents

Executive summary . . . 3
Controls tested . . . 4
Overview and checks performed . . . 7
- Web App Scan — Crawl & Attack . . . 7
- SSL/TLS Check . . . 7
- Email Security Check . . . 7
- HTTP Security Headers . . . 7

What these checks are . . . 7
Findings . . . 7
- Critical (1) . . . 7
- High (3) . . . 9
- Medium (4) . . . 10
- Low (7) . . . 11
- Informational (1) . . . 13

Technical detail . . . 13
- Web App Scan — Crawl & Attack . . . 13
- SSL/TLS Check . . . 13
- Email Security Check . . . 15

Reference: what each check is . . . 16

C

Overall grade C (50/100)

Weighted average of 4 security checks — the latest completed run of each. The score is the same average on a 0–100 scale, so a month that fixes four issues without changing the letter still shows movement.

Controls tested

40	13	4	5
pass	need attention	failing	not checked

40 of 62 controls pass. Every control we tested is listed on the next page with what we saw, including the ones that passed — a report that only lists problems cannot tell you the difference between a control that is healthy and one nobody looked at.

What to deal with first

Most severe first. The CWE reference is the industry identifier for the class of weakness, for pasting into a ticket or mapping to a control framework.

- **SQL Injection — Input is placed directly into a database query**
  CRITICAL · CWE-89 · confirmed
- **Reflected XSS — Input is reflected with the characters needed to inject markup**
  HIGH · CWE-79 · confirmed
- **Directory Listing — Anyone can browse the files in one of your folders**
  HIGH · CWE-548 · inferred
- **DMARC Policy Not Enforced — Impersonation is being watched, not blocked**
  HIGH · CWE-290 · inferred
- **Content Security Policy Allows Inline Script — Your content rules allow scripts written directly into the page**
  MEDIUM · CWE-693 · confirmed
- **Expiring Certificate — Your certificate expires in 16 days**
  MEDIUM · CWE-298 · confirmed

What this covers

This report covers the checks that have been run on this asset: Web App Scan — Crawl & Attack, SSL/TLS Check, Email Security Check, HTTP Security Headers. A check that has never been run on this asset does not appear here at all, so an empty area means untested rather than clean.

Every control this asset's checks tested, with what we saw. A control that passed is listed for the same reason as one that failed: without it, the absence of a problem is indistinguishable from the absence of a check.

Web application

25 of 37 pass

Control	Verdict	What we saw
Transport Encryption — The application answers only over an encrypted connection	Pass	every page we read came back over https
Mixed Content — No encrypted page pulls in unencrypted content	Pass	no page mixed plain http resources into an https page
Security Headers — Security headers are set, on every page rather than on one	Attention	Missing security header: X-Content-Type-Options
Content Security Policy — The content policy restricts what a page may run	Attention	Your content rules allow scripts written directly into the page
Cookie Security Attributes — Cookies carry their protections	Attention	The cookie PHPSESSID can be read by scripts in the page
Session Token in URL — No session token travels in a web address	Pass	no address carried something that identifies a session
Cleartext Password Submission — No password is submitted where it can be read or logged	Pass	every password field posted over https and in the body rather than the address
Cacheable Session Response — Pages that sign someone in are kept out of shared caches	Pass	no response we read set a session without keeping it out of shared caches
Password Field Autocomplete — Password boxes tell the browser not to store what is typed	Pass	no password box we read was left free to be remembered and filled in again
Cross-Site Request Forgery — Forms that change something carry an anti-forgery token	Pass	every form that changes something carried a token another site cannot guess
CSRF Origin Validation — A form cannot be submitted from another site	Pass	no form accepted a submission claiming to come from somewhere else
Default Credentials — No password that ships with the software still works	Not checked	none of the forms the crawl found could be tried with a default password
Object Level Authorization — One account cannot read another account's records	Not checked	this scan was not given two accounts to compare
Function Level Authorization — An ordinary account cannot reach an administrator's pages	Not checked	this scan was not given two accounts to compare
Cross-Domain Form Submission — No form posts to another company's server	Pass	every form posted back to a host you control
Cross-Origin Resource Sharing — Only the sites you chose may read your pages	Pass	no answer to a request claiming another origin opened the page to that origin
JSON Web Token Handling — A sign-in token is signed, strong, and expires	Pass	no sign-in token we were given was unsigned, signed with a guessable key or left without an expiry
JSON Web Token Signature Verification — A forged sign-in token is refused	Not checked	this scan had no sign-in token to test, or found nothing behind a login to test one against
GraphQL Introspection — No GraphQL interface describes itself to strangers	Pass	no interface we found would hand over its schema
Directory Listing — No folder lists its contents	Fail	Anyone can browse the files in one of your folders
Backup and Configuration Files — Nothing is served that was not meant to be	Pass	none of the conventional addresses for a configuration file, a backup or a repository answered
Information Disclosure — Responses do not name files, versions or paths	Attention	This application announces the exact software and versions it runs

Control	Verdict	What we saw
security.txt — A published security contact is current	Pass	the published contact has not expired
DOM-based XSS — No page builds itself out of a value the browser carries	Pass	no script we read takes part of the address, the referrer or the window name into the page or into a redirect
Cross-Site Scripting — Input is escaped where it reaches the page	Fail	Input is reflected with the characters needed to inject markup
Injection — Input does not reach the database, the shell, a template or eval	Fail	Input is placed directly into a database query
Input Validation — Values the application did not expect are handled	Pass	a number outside any plausible range and an empty value were answered rather than failed on
Path Traversal — Files cannot be read off the server's disk	Pass	no path we sent reached a file outside the application
Open Redirect — A redirect cannot be pointed at another site	Pass	no redirect followed an address we supplied
Server-Side Request Forgery — The server cannot be made to fetch an address of our choosing	Pass	nothing we sent made the application look up or fetch a host we control
XML External Entity — XML we send is parsed without following what it points at	Pass	no XML we sent made the parser fetch an entity from outside the document
Deserialization of Untrusted Data — Data from the request is not rebuilt into objects	Pass	nothing we sent was deserialized into a live object by the application
Host Header Injection — The address the application answers for is not trusted blindly	Pass	the application did not act on a host header we chose
HTTP Response Splitting — A request value cannot become a response header	Pass	nothing we sent ended a header line and started one of our own
HTTP Methods Allowed — The server refuses request methods it has no use for	Pass	it refused the method that echoes a request straight back, and it advertises none of the write methods a web server has no use for
Published API Definition — An API description is published deliberately, if at all	Pass	no machine-readable API description answered at a conventional address
Unrestricted File Upload — An upload is restricted, and what is uploaded is not served back	Not checked	no form the crawl could test took a file, so nothing was uploaded

Certificates and transport

8 of 9 pass

Control	Verdict	What we saw
Certificate Validity — Certificate is in date	Attention	expires in 16 days
Certificate Hostname Validation — Certificate matches the hostname	Pass	the name on the certificate covers harbourline.example
Certificate Chain of Trust — Certificate chain is trusted	Pass	issued by Example Trust CA R4
CA-Issued Certificate — Certificate is not self-signed	Pass	issued by Example Trust CA R4
Key Strength and Signature Algorithm — Key and signature are strong enough	Pass	ECDSA 256-bit, ECDSA-SHA256
TLS 1.3 — Modern TLS is offered	Pass	TLS 1.3 offered
Deprecated TLS Versions — Obsolete TLS versions are off	Pass	nothing older than TLS 1.2 is offered
Forward Secrecy — Past traffic stays private if the key is stolen	Pass	6 encryption methods are offered, and all of them keep past traffic private
Known TLS Vulnerabilities — No known TLS weaknesses	Pass	8 checks run, none matched

Email authentication

2 of 8 pass

Control	Verdict	What we saw
SPF — SPF record published	Attention	v=spf1 include:spf.mail.example.net ~all
SPF Hard Fail — SPF refuses unlisted senders	Attention	ends in ~all — unlisted senders are still delivered
DKIM — DKIM signing key published	Pass	key at selector selector1
DMARC — DMARC record published	Attention	v=DMARC1; p=none; rua=mailto:dmarc@harbourline.example
DMARC Policy — DMARC blocks impersonation	Fail	p=none — forged mail is delivered
DMARC Aggregate Reporting — DMARC reports collected	Pass	reports go to mailto:dmarc@harbourline.example
MTA-STS — Encrypted mail delivery is required	Attention	no MTA-STS policy published
TLS-RPT — Failed mail encryption is reported	Attention	no TLS-RPT record

Security headers

5 of 8 pass

Control	Verdict	What we saw
Strict-Transport-Security	Pass	max-age=31536000
Content-Security-Policy	Pass	default-src 'self'; script-src 'self' 'unsafe-inline'; frame-ancestors 'self'
X-Frame-Options	Pass	SAMEORIGIN
X-Content-Type-Options	Attention	Missing.
Referrer-Policy	Attention	Missing.
Permissions-Policy	Pass	camera=(), microphone=(), geolocation=()
X-XSS-Protection	Pass	Not set (or disabled), which is correct — it is deprecated.
Server	Attention	nginx/1.24.0

harbourline.example

Latest result of 4 security checks | 17 October 2026

C

Overall grade
Average across the 4 checks below — the latest result of each type.

Checks performed

F **Web App Scan — Crawl & Attack**
Input on the shipment tracking page reaches the database unescaped.

B **SSL/TLS Check**
The certificate is valid and the protocols are current, but it expires in 16 days.

C **Email Security Check**
Anyone can send email that looks like it came from this domain and receivers are asked to deliver it anyway.

C **HTTP Security Headers**
4 of 6 core security headers are set; X-Content-Type-Options and Referrer-Policy are missing.

What these checks are

One line each. The full description of every check, and how to read its grade, is at the end of this report.

Web App Scan — Crawl & Attack
Everything the crawl does, and then we test what it found by sending it crafted values.

SSL/TLS Check
The certificate this server presents, and the encryption it agrees to use.

Email Security Check
The published DNS records that decide who may send email as this domain.

HTTP Security Headers
The security instructions this site sends to a browser with each page.

1	3	4	7	1
CRITICAL	HIGH	MEDIUM	LOW	INFO

Findings (16)

Critical (1)

CRITICAL Web App Scan — Crawl & Attack · CWE-89

SQL Injection — Input is placed directly into a database query

A value this page accepts is put straight into a database query instead of being passed to it separately. We sent a single harmless character — a quote — and MySQL or MariaDB answered with a syntax error; the

same request without that one character came back as an ordinary page, which is what makes the error ours rather than the page's own. Anyone who can reach this page can therefore change what the query asks for: read records they are not entitled to, alter them, or delete them. We stopped at proving the parameter reaches the database and did not run any query of our own.

For technical readers

A value from the request is concatenated into a SQL statement, so the database parses the attacker's punctuation as syntax rather than reading it as data — which reaches every row that connection can read, and on several engines the file system and command execution as well. The repair is parameterised statements everywhere, including the places that tempt people to build strings: an ORDER BY column or a table name cannot be bound as a parameter and needs an allowlist of permitted values instead. Escaping routines and request filtering reduce the surface without closing it.

What we sent, and what came back

The request with one quote added to the tracking reference

```
GET /track?ref=HL-20931' HTTP/1.1
Host: harbourline.example
User-Agent: Mozilla/5.0 (ClickScan)
```

```
HTTP/1.1 500 Internal Server Error
Content-Type: text/html

<h1>Error</h1><p>You have an error in your SQL syntax; check the manual that corresponds to your MySQL
server version for the right syntax to use near ''HL-20931''' at line 1</p>
```

The same request without the quote, for comparison

```
GET /track?ref=HL-20931 HTTP/1.1
Host: harbourline.example
User-Agent: Mozilla/5.0 (ClickScan)
```

```
HTTP/1.1 200 OK
Content-Type: text/html

<h1>Shipment HL-20931</h1><p>Status: In transit, Rotterdam</p>
```

What to do

Stop building this query by joining text together. Use the database library's parameter binding — prepared statements, bound or named parameters — so values are sent to the database separately from the instruction and can never change its meaning. Escaping quotes is not an adequate substitute and fails on the numeric and identifier cases. Then check whether the same pattern is used elsewhere in the application, because it is rarely in one place, and turn off detailed database errors in production so the next defect does not describe itself to a stranger.

High (3)

HIGH Web App Scan — Crawl & Attack · CWE-79

Reflected XSS — Input is reflected with the characters needed to inject markup

A value this page accepts is sent straight back into the page with the punctuation an attacker needs left intact. We submitted four harmless characters — a less-than sign — and they came back exactly as sent, in the page's text, where they can start a new tag, rather than being converted into their harmless display form. That is what makes a crafted link dangerous: someone who sends your visitor a link with script in it has it run inside your site, as that visitor, with their session — which is enough to read what they can read or act as them. We proved the characters survive and deliberately did not send any script.

For technical readers

Input from the request is returned in the response without being encoded for the context it lands in, so a crafted value stops being data and becomes markup the browser parses — script then runs in this origin's security context, with whatever the signed-in session can reach. The repair is output encoding chosen by SINK rather than by input: HTML body, attribute value, JavaScript string and URL each need a different one, applied where the value is rendered. A Content-Security-Policy that forbids inline script limits what an injection achieves and is worth having, but it is defence in depth, not the fix.

What we sent, and what came back

Our marker, sent in the search box

```
GET /search?q=cs7f3e%3C%22%27%3E HTTP/1.1
Host: harbourline.example
HTTP/1.1 200 OK
Content-Type: text/html

<h2>Results for cs7f3e<"'></h2>
```

What to do

Encode values on the way OUT, according to where they are being put: HTML-encode for page text, attribute-encode inside a tag, and use the framework's own escaping rather than hand-written replacement. Most template engines do this by default and are usually switched off for the field in question, so look for the "raw" or "unescaped" marker on this value first. A Content-Security-Policy limits the damage and is not a substitute for encoding.

HIGH Web App Scan — Crawl & Attack · CWE-548

Directory Listing — Anyone can browse the files in one of your folders

Opening https://harbourline.example/uploads/ returns an automatic list of the files in that folder rather than a page. Whoever looks can read every filename there, including backups, exports and anything else left behind that was never meant to be found.

For technical readers

The server renders an index of the directory instead of refusing the request, so every file in it can be enumerated without guessing a single name — which matters most for the files nothing links to, because that is where backups, exports and configuration collect. The repair is to switch auto-indexing off; removing the files matters more than removing the listing, which only made them easy to find.

What to do

Switch off automatic directory listing for this folder on your web server, and move anything in it that should not be public.

HIGH Email Security Check · CWE-290

DMARC Policy Not Enforced — Impersonation is being watched, not blocked

Your DMARC policy is set to "none". That collects reports about who is sending as your domain, but asks receivers to deliver the mail anyway — including mail from people impersonating you. It is the first stage of a rollout, not the finished article.

What to do

Once the DMARC reports show only senders you recognise, change p=none to p=quarantine, then to p=reject. Each step is a one-line DNS change and can be reversed instantly.

Medium (4)

MEDIUM Web App Scan — Crawl & Attack · CWE-693

Content Security Policy Allows Inline Script — Your content rules allow scripts written directly into the page

The script rule allows inline script. That is the exact thing content rules exist to stop: an attacker who can place markup on the page can place a working script with it, and the browser will run it.

What we sent, and what came back

The policy every page is served with

```
GET / HTTP/1.1
Host: harbourline.example
HTTP/1.1 200 OK
Content-Security-Policy: default-src 'self'; script-src 'self' 'unsafe-inline'; frame-ancestors 'self'
```

What to do

Remove 'unsafe-inline' and give each of your own inline scripts a nonce or a hash. A browser ignores 'unsafe-inline' once a nonce or hash is present, so you can keep it for very old browsers without weakening the rule.

MEDIUM SSL/TLS Check · CWE-298

Expiring Certificate — Your certificate expires in 16 days

It's still valid, but expiry is approaching.

What to do

Schedule a renewal or confirm auto-renewal is enabled.

MEDIUM Email Security Check · CWE-290

SPF Soft Fail — Your SPF record does not ask for impersonated mail to be refused

SPF is published, but it ends in ~all, which asks receivers to accept a message from an unlisted server and only mark it.

What to do

End the record with -all once you are sure every legitimate sender is listed.

MEDIUM Email Security Check · CWE-319

Missing MTA-STS — Mail to this domain can be delivered unencrypted

MTA-STS is not set up. It is what tells other mail servers that they must use encryption when delivering to you, and must refuse to deliver if encryption cannot be negotiated. Without it, an attacker positioned between two mail servers can strip the encryption and read the messages in transit.

What to do

Publish an MTA-STS policy: a TXT record at _mta-sts.harbourline.example and a policy file at https://mta-sts.harbourline.example/.well-known/mta-sts.txt. Start in testing mode, then switch to enforce.

Low (7)

LOW Web App Scan — Crawl & Attack · CWE-1004

Cookie Without HttpOnly Flag — The cookie PHPSESSID can be read by scripts in the page

Your site set a cookie named PHPSESSID without marking it HttpOnly, so any JavaScript running in the page can read it. If somebody ever gets a script onto one of your pages — through a comment box, an advert or a compromised third-party file — they can take that cookie and use it as the visitor.

For technical readers

The cookie is readable from JavaScript, so any script running on the page can read it and send it elsewhere — including one injected through cross-site scripting, and one shipped by a dependency that was compromised upstream. The repair is `HttpOnly` on every cookie the page's own code does not need to read, which for a session cookie is always.

What to do

Add the HttpOnly attribute when the cookie is set. Only leave it off for a cookie your own scripts genuinely have to read.

LOW Web App Scan — Crawl & Attack · CWE-693

Missing Security Header — Missing security header: X-Content-Type-Options

Without this header some browsers guess ("sniff") a response's type and may run a file as script that was never meant to be. We read 151 pages and none of them carried X-Content-Type-Options.

For technical readers

A response header the browser would have used to constrain the page is absent, so a defence that costs nothing to deploy is not in place. These headers rarely stop an attack on their own; what they do is limit what an injection or a framing attack ACHIEVES, which is why they matter most on the day something else goes wrong. The repair is to set them at the edge, so every response carries them rather than each route remembering.

What to do

Add X-Content-Type-Options: nosniff.

LOW Web App Scan — Crawl & Attack · CWE-693

Missing Security Header — Missing security header: Referrer-Policy

Without a Referrer-Policy the full URL of your pages (which can carry tokens or private paths) is sent to every third-party site your pages link to or load from. We read 151 pages and none of them carried Referrer-Policy.

For technical readers

A response header the browser would have used to constrain the page is absent, so a defence that costs nothing to deploy is not in place. These headers rarely stop an attack on their own; what they do is limit what an injection or a framing attack ACHIEVES, which is why they matter most on the day something else goes wrong. The repair is to set them at the edge, so every response carries them rather than each route remembering.

What to do

Add Referrer-Policy: strict-origin-when-cross-origin (or stricter).

LOW Email Security Check · CWE-16

Missing TLS-RPT — No reports when encrypted delivery fails

TLS-RPT is not set up, so if another provider cannot deliver mail to you securely, nobody is told. Delivery problems of this kind are otherwise invisible until a customer says they never received something.

What to do

Add a TXT record at _smtp._tls.harbourline.example — "v=TLSRPTv1; rua=mailto:tls@harbourline.example".

LOW HTTP Security Headers · CWE-693

Missing Security Header — Missing security header: X-Content-Type-Options

Without this header some browsers guess ("sniff") a response's type and may run a file as script that was never meant to be.

For technical readers

A response header the browser would have used to constrain the page is absent, so a defence that costs nothing to deploy is not in place. These headers rarely stop an attack on their own; what they do is limit what an injection or a framing attack ACHIEVES, which is why they matter most on the day something else goes wrong. The repair is to set them at the edge, so every response carries them rather than each route remembering.

What to do

Add X-Content-Type-Options: nosniff.

LOW HTTP Security Headers · CWE-693

Missing Security Header — Missing security header: Referrer-Policy

Without a Referrer-Policy the full URL of your pages (which can carry tokens or private paths) is sent to every third-party site your pages link to or load from.

For technical readers

A response header the browser would have used to constrain the page is absent, so a defence that costs nothing to deploy is not in place. These headers rarely stop an attack on their own; what they do is limit what an injection or a framing attack ACHIEVES, which is why they matter most on the day something else goes wrong. The repair is to set them at the edge, so every response carries them rather than each route remembering.

What to do

Add Referrer-Policy: strict-origin-when-cross-origin (or stricter).

LOW HTTP Security Headers · CWE-200

Software Version Disclosure — Server discloses software: Server

The Server header is advertising the exact software and version you run, which helps an attacker match it to known exploits.

For technical readers

The response names the product and its version, which is not an exposure in itself but removes the reconnaissance step: an attacker matches the version against published advisories rather than probing to find out. The repair is to suppress the version in responses — and to read the finding as a prompt to check whether the named version is current, which is usually the more important question.

What to do

Remove the version (or the whole Server header) at your web server or reverse proxy.

Informational (1)

INFO Web App Scan — Crawl & Attack · CWE-200

Software Version Disclosure — This application announces the exact software and versions it runs

Responses name nginx 1.24.0, PHP. We read that from what the application sent us, unprompted. An exact version turns "is anything here exploitable" into a lookup — somebody matches the version against published vulnerabilities for it instead of probing to find out.

For technical readers

The response names the product and its version, which is not an exposure in itself but removes the reconnaissance step: an attacker matches the version against published advisories rather than probing to find out. The repair is to suppress the version in responses — and to read the finding as a prompt to check whether the named version is current, which is usually the more important question.

What to do

Stop sending the version. On Apache set ServerTokens Prod and ServerSignature Off; on nginx set server_tokens off; for PHP set expose_php = Off. If a proxy or CDN sits in front, strip the headers there. Then check that the versions you are running are still supported and patched: hiding the banner is not the same as being up to date.

Web application

Mapped from https://harbourline.example/

What this scan covered

We found 212 addresses and fetched 164 of them.
We left 6 outside the scope set for this target, 8 on another host and 22 that are a kind of file a crawl does not follow.
2 addresses did not answer, so we have nothing to report about them.
10 addresses were found and never fetched. Of those, 10 were queued when the crawl stopped.

Application surface

37 places a value can be inserted - parameters, form fields, headers and path segments - were found.
It links to 1 host we did not follow: fonts.cdn.example.

What it looks like it is running

PHP (language) - a cookie named PHPSESSID
nginx 1.24.0 (server) - Server: nginx/1.24.0
Each of these is inferred from what the application volunteers - a header, a cookie name, a script path - rather than established by a probe.

Full scan detail

The complete technical data behind the findings above — the same detail the app shows in its expandable panels. Use your PDF viewer's outline to jump between sections.

Resolved addresses

Address tested	`203.0.113.40`

Certificate details

The certificate the server presented, and how it validated.

Issued for	harbourline.example
Issued by	Example Trust CA R4
Valid from	1 August 2026
Valid until	30 October 2026
Days until expiry	16
Expired	No
Hostname match	Matches the certificate
Self-signed	No
Chain trusted	Trusted to a known root
Public key	ECDSA 256-bit
Signature algorithm	ECDSA-SHA256
Negotiated protocol	TLS 1.3
Negotiated cipher	TLS_AES_128_GCM_SHA256
Protocol support	SSL 3.0 off, TLS 1.0 off, TLS 1.1 off, TLS 1.2 on, TLS 1.3 on
OCSP stapling	Stapled in the handshake
ALPN (application protocol)	h2

Subject alternative names

`harbourline.example, www.harbourline.example, portal.harbourline.example`

Named-weakness checks

Each known weakness we probe for. "Not tested" means we could not get a definite answer on the wire — it is not a pass.

OK **Heartbleed (CVE-2014-0160)**
The server did not return memory to a malformed heartbeat.

OK **CCS injection (CVE-2014-0224)**
An early ChangeCipherSpec was refused.

OK **ROBOT (Bleichenbacher oracle)**
No RSA key exchange is offered.

OK **POODLE (SSLv3)**
SSL 3.0 is refused.

OK **DROWN (SSLv2)**
SSLv2 is refused.

OK **Secure renegotiation (RFC 5746)**
Supported.

OK **CRIME (TLS compression)**
Compression is off.

OK **Weak Diffie-Hellman / Logjam**

No finite-field Diffie-Hellman suite is offered.

Accepted cipher suites

Suites the server accepted from us, one row per protocol version — weakest first. The DHE, ARIA, Camellia and SEED families are offered as well as the common ones, so a family missing here is one the server did not accept. It is still not every suite that exists, so a very unusual one may not appear. TLS 1.3 is the one exception to listing only what was demonstrated — its three suites cannot be offered individually, so all three are listed whenever TLS 1.3 handshakes.

6 strong

Version	Cipher suite	Strength	Fwd secrecy
TLS 1.2	`TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`	STRONG	Yes
TLS 1.2	`TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`	STRONG	Yes
TLS 1.2	`TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256`	STRONG	Yes
TLS 1.3	`TLS_AES_128_GCM_SHA256`	STRONG	Yes
TLS 1.3	`TLS_AES_256_GCM_SHA384`	STRONG	Yes
TLS 1.3	`TLS_CHACHA20_POLY1305_SHA256`	STRONG	Yes

Email security detail

What each published record says, and the state we found it in. "Not checked" means we could not establish an answer — it is not a pass. The raw lookups behind all of it follow this section.

Mail servers

2 mail servers published for harbourline.example, in the order sending servers should try them.

Pri 10 **`mx1.mail.example.net`**
198.51.100.10

Pri 20 **`mx2.mail.example.net`**
198.51.100.11

Published records

SPF, DKIM and DMARC are what stop someone sending email in this domain's name. MTA-STS and TLS-RPT protect mail in transit; BIMI is branding and carries no security weight.

WARN **SPF**

ends in ~all (accept anything else but mark it) · 2 of the 10 permitted DNS lookups used · authorises 1 other sender

```
v=spf1 include:spf.mail.example.net ~all
```

- it ends in ~all, which asks receivers to accept a message from an unlisted server and only mark it.

PASS **DKIM**

signing keys found: selector1 (rsa 2048-bit) · 6 common selector names tried; selectors cannot be listed from DNS, so a key under a name we did not guess would not be found

WARN **DMARC**

policy p=none (forged mail is delivered normally; this only monitors) · aggregate reports requested · alignment adkim=r aspf=r

```
v=DMARC1; p=none; rua=mailto:dmarc@harbourline.example
```

NOT SET **MTA-STS**

NOT SET **TLS-RPT**

NOT SET **BIMI**

Technical addendum: DNS responses

Exactly what was asked of DNS and what came back — the evidence behind every check in this report. Your DNS administrator can verify each record here without taking our word for it.

MX harbourline.example

```
10 mx1.mail.example.net
20 mx2.mail.example.net
```

TXT harbourline.example

```
v=spf1 include:spf.mail.example.net ~all
```

TXT _dmarc.harbourline.example

```
v=DMARC1; p=none; rua=mailto:dmarc@harbourline.example
```

TXT selector1._domainkey.harbourline.example

```
v=DKIM1; k=rsa; p=MIIBIjANBg...
```

Reference: what these standards mean

SPF — Sender Policy Framework

SPF is a DNS record listing the servers allowed to send mail for your domain. A receiving server checks it against the sending address; an unlisted sender fails, and the message may be rejected or filtered. The whole check may use at most 10 DNS lookups — each include:, a, mx, ptr and exists: costs one, and so do the providers you include. Exceed the limit and the record fails permanently, at which point most receivers treat the domain as having no SPF at all.

DKIM — DomainKeys Identified Mail

DKIM adds a cryptographic signature to each message. Receivers fetch the public key from DNS at selector._domainkey.yourdomain and verify it. A pass proves the message was not altered in transit and that the sending domain takes responsibility for it. There is no way to discover which selector a domain uses — DNS will confirm whether a specific name exists but will not list them — so an automated check can only try the common names, and a miss is never proof that DKIM is absent.

DMARC — Domain-based Message Authentication

DMARC tells receivers how to handle mail that fails SPF and DKIM. p=none only collects reports; p=quarantine sends forgeries to spam; p=reject refuses them outright. The usual path is none, then quarantine, then reject, reading the reports at each step. The rua= address is what makes those reports exist; enforcement without reporting is acting with the lights off.

MTA-STS — Strict Transport Security

MTA-STS requires TLS, with valid certificates, for mail delivered to you. It has two halves: a DNS TXT record at _mta-sts.yourdomain, and a policy file served over HTTPS at https://mta-sts.yourdomain/.well-known/mta-sts.txt. Both must work. A record advertising a policy that cannot be fetched is worse than publishing neither, because senders that cached it keep enforcing it.

TLS-RPT — SMTP TLS Reporting

TLS-RPT is a DNS record telling receiving servers where to send reports about your domain's TLS delivery results. It surfaces certificate problems, downgrade attacks and failed handshakes without exposing message content. Publish it at _smtp._tls.yourdomain, alongside MTA-STS.

BIMI — Brand Indicators for Message Identification

BIMI lets participating mailbox providers show your logo beside authenticated mail. It only works once DMARC is enforced (quarantine or reject), and some providers also require a Verified Mark Certificate. It does not change delivery — fix the authentication underneath before adding it.

Reference: what each check is

For a reader who did not run these checks: what each one examined, why it matters, and how its grade is arrived at.

Web App Scan — Crawl & Attack

What we checked

We do the whole of the crawling scan first — follow the links, record every address, parameter and form, and read what comes back. Then we go back to what we found and TEST it, by submitting values chosen to make a badly built application misbehave. We try to make a database answer a question it was never asked, to get a command run on the server, to get our own markup rendered back to another visitor, to read a file outside the application's own folder, and to make the application fetch an address of our choosing. We try published default passwords on the sign-in forms we are able to submit to — at most three per account name, so we cannot lock out a real administrator. A form that carries an anti-forgery token is left alone and reported as NOT TESTED rather than as clean, and on a modern framework that is most login forms. Where you have given us two sign-ins AND told us which of them is the higher-privileged one, we ask for the same pages as each and compare the answers. Two accounts you have not ranked, or two at the same level, are not compared at all — we will not invent an order between accounts you consider equal — and the report says so rather than reporting nothing found. Two things follow from all that and are said plainly rather than buried. THIS SCAN CHANGES DATA on the application it is pointed at: where a form stores what it is given, we store an inert marker in it, and the report lists the places a marking rule submitted to. One gap in that list we would rather state than have you discover — a file left by the upload test is not yet enumerated there. And we deliberately submit NOTHING to forms that look destructive — a delete, a cancel, a reset. Those stay on the map and the report counts them; they are simply never given a value of ours. Point this at a target you are ready to have tested.

Why it matters

Reading an application tells you what it exposes. Only submitting to it tells you whether it mishandles what it is given — and that is where the breaches come from. A form that passes what you type straight into a database query, a page that renders another visitor's markup, an address that hands over another customer's record because nothing checked whose it was: none of these is visible from the outside, and none of them appears in a scan that only looks. They are also the findings that matter most, because each one is a working way in rather than a weakness somebody might combine with something else.

How to read the grade

The letter is driven by the worst finding in the run, and the report names which one drove it. Everything the ATTACKING pass claims, it demonstrates — a value that came back rendered, a database error that appears only in the tampered request, an address that answered for an account that should not have reached it — so an attacking finding is not a suspicion we are passing on to you. Be aware that this run also carries the whole of the crawling scan's reading, and a few of those are inferred rather than demonstrated, so the letter can be driven by one of them; every finding says which kind it is. An A means everything this scan was able to test answered the way it should. It does not mean the application is safe: what we could not test is reported as not tested rather than counted as a pass, which is why a run that stopped at its budget, a sign-in we could not complete, or a target with only one account says so in as many words. Checks that need our out-of-band listener report that they were not checked when it is not configured, rather than reporting clean.

SSL/TLS Check

What we checked

We connected to this server the way a browser does and examined the certificate it presented and the encryption it agreed to use: who issued the certificate, whether it is still valid and matches the name, which protocol versions the server still accepts, and which encryption ciphers it will agree to. We also probed for a set of named weaknesses by name.

Why it matters

A certificate that has expired or does not match the name makes browsers warn your visitors, and most of them will leave. Obsolete protocols and broken ciphers are worse and quieter: the padlock still appears, while somebody positioned between your visitor and your server can read or alter what passes.

How to read the grade

The grade follows the most serious problem found, not an average. Genuinely broken encryption is an F here — a padlock that means nothing is not a C. A clean setup that also tells browsers to refuse plain HTTP for at least six months earns A+.

Email Security Check

What we checked

We read the public DNS records that govern who is allowed to send email using this domain name — SPF, DKIM and DMARC — along with the mail servers it publishes. Nothing was sent and no mailbox was touched.

Why it matters

These three records are what stops anyone on the internet putting your domain in the From line of an email. Without them, a message that appears to come from your company can be delivered to your customers, your staff or your suppliers, and nothing in the message will look wrong to them. This is how most invoice fraud begins.

How to read the grade

Each record contributes points, weighted by how much protection it actually provides — a DMARC policy set to take no action scores far below one that rejects. The grade explanation names the heaviest deductions.

HTTP Security Headers

What we checked

We requested one page over HTTPS and read the security instructions the server sent back with it. These headers tell a browser what it may and may not do with your site — whether to allow it inside a frame, where scripts may be loaded from, whether to guess at file types.

Why it matters

Without them a browser falls back to permissive defaults. That is what makes clickjacking possible (your page loaded invisibly inside somebody else's), and it removes the last line of defence against a script injected into your page. It also shows when a server volunteers its exact software version, which tells an attacker which known flaws to try first.

How to read the grade

The grade is driven by how many of the six core protection headers are present and correct. The additional headers and the information-disclosure checks are reported but do not by themselves move the letter.

Report generated 17 October 2026, 19:36 UTC for harbourline.example.