SECURITY CASE STUDY

Portfolio Website Security Audit

An authorized, sanitized self-assessment of **thomasmoran.co**, focused on practical web hardening, the Ask Thomas AI request path, and responsible public documentation.

JULY 2026	4	3	1
Assessment date	Findings recorded	Resolved or mitigated	Open limitation

DOCUMENT STATUS

Public redacted report

This document is a learning artifact and portfolio case study. It is not an independent certification, formal compliance audit, or third-party penetration-test attestation. Sensitive configuration and actionable exploit detail are intentionally excluded.

1. Executive summary

The review identified opportunities to strengthen browser-facing protections and reduce abuse of the portfolio's AI endpoints and local import workflows. Remediation added layered transport and response-header controls, bounded same-origin requests, allowlisted AI payloads, local file validation, export neutralization, and database-backed rate limiting. The resulting design reduces common web risk while preserving a small, understandable attack surface.

2. Scope and authorization

Testing was authorized because the assessed application is my own portfolio. Scope was limited to the public website at thomasmoran.co and first-party application routes under my control.

- Included: HTTPS behavior, response-header configuration, public information exposure, AI request validation, local file imports, generated exports, rate limiting, and source-level control review.
- Excluded: denial-of-service activity, social engineering, destructive testing, credential attacks, third-party infrastructure, and systems not owned by the assessor.

3. Methodology

- Defined authorized boundaries and documented exclusions before testing.
- Reviewed transport behavior and browser-facing response protections.
- Traced both AI request paths and reviewed local CSV/JSON import and export handling.
- Ranked observations by practical likelihood and impact in this portfolio context.
- Implemented remediation, rebuilt the application, and verified production configuration.

4. Findings register

ID	Observation	Severity	Status
SEC-01	Missing defense-in-depth response headers	Medium	**Resolved**
SEC-02	AI endpoint abuse and excessive request volume	Medium	**Mitigated**
SEC-03	Sensitive detail in a public audit artifact	Low	**Avoided**
SEC-04	Independent penetration testing not completed	Info	**Open**

SEC-01 | Browser response hardening

Observation: The application needed stronger defense-in-depth controls at the response layer.

Remediation: Added Strict-Transport-Security, Content-Security-Policy, Permissions-Policy, Referrer-Policy, X-Content-Type-Options, and clickjacking protection. HTTP traffic redirects to HTTPS.

Result: Resolved. Browsers receive explicit restrictions that reduce downgrade, framing, MIME-sniffing, referrer leakage, and unapproved resource-loading risk.

SEC-02 | AI endpoint abuse controls

Observation: A public AI endpoint can attract automated traffic, oversized inputs, and repeated requests that consume resources.

Remediation: Enforced same-origin JSON requests, streamed body-size limits, strict question validation, allowlisted AccessGraph rules and evidence, no-store API responses, and database-backed rate limiting. Secrets remain server-side and are not sent to the browser.

Result: Mitigated. The endpoint is more resistant to casual abuse, while monitoring and future tuning remain ongoing operational tasks.

SEC-03 | Public information exposure

Observation: Publishing a detailed report can accidentally reveal sensitive configuration or provide unnecessary attack guidance.

Remediation: Produced this redacted case study and excluded credentials, private values, internal identifiers, and detailed exploit steps.

Result: Avoided. The report demonstrates process and outcomes without increasing practical risk.

5. Controls implemented

Control area	Implementation	Security purpose
Transport	HTTPS redirect and HSTS	Reduce downgrade and unencrypted transport risk
Content loading	Content Security Policy	Restrict resource origins and reduce injection impact
Browser features	Permissions Policy	Limit access to unneeded browser capabilities
Framing	Frame restrictions	Reduce clickjacking exposure
Content types	X-Content-Type-Options	Prevent MIME-type sniffing
Referrers	Referrer Policy	Reduce unnecessary URL information leakage
AI endpoints	Bounded same-origin JSON + D1 limits	Reduce malformed input and automated abuse
Local imports	Type, size, schema + field checks	Reject malformed CSV and JSON data
Generated exports	Formula + Markdown neutralization	Reduce spreadsheet and markup injection risk

RESIDUAL LIMITATION

Independent testing remains open

This self-assessment does not provide the independence, breadth, or assurance of a professional third-party penetration test. Future work includes automated dependency monitoring, repeat testing after major changes, and an external review when appropriate.

6. Lessons learned

- Security controls are strongest when layered across transport, browser behavior, server validation, and abuse prevention.
- A remediation is not complete until the production behavior is verified after deployment.
- Responsible security communication includes knowing which details should not be made public.
- Accurate limitations build more trust than overstating the assurance provided by a self-assessment.

7. Conclusion

The project demonstrates a complete security workflow: define scope, review risk, prioritize findings, implement controls, verify the result, and document residual limitations. The portfolio now provides both a working application and a concise record of the security decisions behind it.

Contact: tmoran532@gmail.com | thomasmoran.co | github.com/Tdogm