htmlguy.

Sample report. The client, site, findings and numbers are made up to show what a report looks like.

COMPLETE REPORT

Website security report

Example Booking App (sample)

Checked	Code: https://git.example.com/booking/app (commit a7d2e9f1c4b8)
Checks	Full check (code review)
Dates	November 4, 2026
Report ID	HG-EXAMPLE-202611
Prepared by	Shane Stebner, HTMLGuy LLC

Confidential. Prepared for Example Booking App (sample) only, under the terms at the end of this report.

HTMLGuy LLC · South Florida · htmlguy.com

The short version

D

At least one serious problem. Fix it soon.

Good progress since last month: the old password was changed and the outdated jQuery is gone. One problem still lets a signed-in customer see other customers' bookings. Fix that first; it takes minutes.

Since the last check: 2 fixed, 1 new.

CRITICAL	HIGH	MEDIUM	LOW	INFO
0	1	0	2	0

Do these first

1. **Signed-in customers can see other customers' bookings** HIGH
   Make the booking page only return bookings that belong to the signed-in customer. (HG-EXAMPLE-0001)
2. **No Content-Security-Policy header** LOW
   Add a Content-Security-Policy header that only allows the site's own scripts and the services it uses. (HG-EXAMPLE-0005)
3. **Error page shows internal details** LOW
   Show a plain error message and keep the details in the server logs. (HG-EXAMPLE-0006)

What's already done well

- Every page uses HTTPS with a valid certificate
- Card payments go through Stripe Checkout, so card numbers never touch the site

What the grade means: A, nothing serious; B, a few things worth fixing; C, several problems; D, at least one serious problem; F, at least one critical problem. Findings marked "possible" aren't counted until they're confirmed.

What we checked

- **Full check of https://git.example.com/booking/app**, commit a7d2e9f1c4b8, on November 4, 2026. A review of the code, its dependencies, configuration and deployment setup, using standard security tools and a line-by-line review of how requests, permissions and data flow. Nothing was run against live systems.

Permission given by Pat Example, owner on 2026-10-01; ownership verified by: Email address at the site's own domain.

Every area below comes from the OWASP Application Security Verification Standard (ASVS 5.0) and the OWASP Top 10, with DNS, email and platform checks added. Each check was marked passed, problem found, not applicable (N/A) or not checked.

Area	Passed	Problems	N/A	Not checked
HTTPS and certificates	6	0	0	0
Browser protections and security headers	8	1	0	0
Front-end code and third-party scripts	10	0	0	0
Exposed files and information leaks	6	0	0	0
Domain, DNS and email	0	0	0	8
CMS, plugins and platform versions	5	0	0	0
Dependencies and supply chain	6	0	0	0
Secrets and credentials	6	0	0	0
Sign-in and accounts	9	0	0	0
Sessions, tokens and single sign-on	6	0	0	0
Permissions and access control	6	1	0	0
Injection and output encoding	10	0	0	0
Input validation and business logic	6	0	0	0
APIs and web services	6	0	0	0
File uploads and file handling	5	0	0	0
Server-side requests	3	0	0	0
Encryption and data protection	4	0	0	0
Personal data and privacy	6	0	0	0
Server and app configuration	4	1	0	0

Area	Passed	Problems	N/A	Not checked
Logging, alerts and error handling	5	0	0	0
Code repository and deployment pipeline	5	0	0	0
Hosting, cloud and infrastructure	6	0	0	0
Database rules	5	0	0	0
AI and LLM features	0	0	6	0
Payments, webhooks and integrations	4	0	0	0
Abuse protection and resilience	4	0	0	0
WebSockets, real-time and WebRTC	0	0	3	0
Secure design and architecture	4	0	0	0

Limits of this check

This is a security review, not a penetration test: nobody tried to break into the live site. It covers the dates and code version shown. No check can promise to find every problem, and new ones appear as software changes. It isn't a certification such as SOC 2 or PCI.

All findings

ID	Finding	Severity	Status	Where	Effort to fix
HG-EXAMPLE-0001	Signed-in customers can see other customers' bookings	HIGH	Still open	app/api/bookings/[id]/route.ts:12	Minutes
HG-EXAMPLE-0005	No Content-Security-Policy header	LOW	Still open	next.config.js:8	Hours
HG-EXAMPLE-0006	Error page shows internal details	LOW	New	app/contact/error.tsx:6	Minutes

From the last check

ID	Finding	Severity	Now
HG-EXAMPLE-0002	Database password left in the code history	HIGH	Fixed
HG-EXAMPLE-0003	Email can be sent pretending to be the company	MEDIUM	Not re-checked
HG-EXAMPLE-0004	Old jQuery with known security bugs	MEDIUM	Fixed

IDs stay the same from one check to the next, so you can track each finding until it's fixed.

HG-EXAMPLE-0001 · PERMISSIONS AND ACCESS CONTROL

Signed-in customers can see other customers' bookings

HIGH | Still open | Confidence: confirmed | Effort: Minutes

What's wrong: The page that shows a booking checks that someone is signed in, but not that the booking is theirs.

Why it matters: Any customer could change the booking number in the address bar and see another customer's name, phone number and booking details.

What to do: Make the booking page only return bookings that belong to the signed-in customer.

Technical detail

GET /api/bookings/[id] loads the booking by primary key without filtering by the session user.

Impact: Disclosure of every customer's name, address, phone and booking history to any account holder.

CVSS 4.0	7.1 `CVSS:4.0/AV:N/AC:L/AT:N/PR:L/UI:N/VC:H/VI:N/VA:N/SC:N/SI:N/SA:N`
CWE	CWE-639
References	ASVS 5.0 V8.2.2, OWASP A01:2025, OWASP API1:2023

Where to look

- `app/api/bookings/[id]/route.ts:12–15 in GET`

Evidence

CODE

```
const booking = await db.booking.findUnique({ where: { id: params.id } })
if (!session) return unauthorized()
return Response.json(booking)
```

How to see it

With two local test accounts, sign in as A and request /api/bookings/<a booking id of B>; the booking is returned.

How to fix it

Filter the booking lookup by the signed-in user and return 404 otherwise.

1. Look the booking up by id and customer id together.
2. Return 404 when nothing matches.
3. Add a test that one customer cannot read another's booking.

Exact change

`app/api/bookings/[id]/route.ts:12`: Only the owner's booking can be found; anyone else gets the same answer as a missing booking.

BEFORE

```
const booking = await db.booking.findUnique({ where: { id: params.id } })
```

AFTER

```
const booking = await db.booking.findFirst({ where: { id: params.id, customerId:
session.user.id } })
if (!booking) return notFound()
```

How to confirm it's fixed: A test signs in as one customer, requests another customer's booking, and gets 404.

A ready-to-paste fix prompt for your AI coding tool is in the fix pack under HG-EXAMPLE-0001.

HG-EXAMPLE-0005 · BROWSER PROTECTIONS AND SECURITY HEADERS

No Content-Security-Policy header

LOW | Still open | Confidence: confirmed | Effort: Hours

What's wrong: The site doesn't tell browsers which scripts are allowed to run on its pages.

Why it matters: If someone ever slips a bad script onto a page, browsers will run it. This setting would block most of those.

What to do: Add a Content-Security-Policy header that only allows the site's own scripts and the services it uses.

Technical detail

Responses from https://example.com/ include no Content-Security-Policy header.

Impact: No defense in depth against XSS or injected third-party scripts.

CVSS 4.0	2.1 `CVSS:4.0/AV:N/AC:H/AT:P/PR:N/UI:A/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N`
CWE	CWE-693
References	ASVS 5.0 V3.4.3, OWASP A02:2025

Where to look

- `next.config.js:8 in headers`

Evidence

CODE

```
next.config.js: headers() returns only Strict-Transport-Security
```

How to see it

curl -sI https://example.com/ | grep -i content-security-policy (returns nothing)

How to fix it

Send a Content-Security-Policy header from the hosting config.

1. List the domains the site loads scripts, styles and images from.
2. Add a CSP allowing only those, in report-only mode first.

3. Switch to enforcing once no reports appear for a week.

Exact change

`next.config.js:8`: Adds the policy next to the existing HSTS header.

BEFORE

```
headers: [{ key: 'Strict-Transport-Security', value: 'max-age=63072000' }]
```

AFTER

```
headers: [
  { key: 'Strict-Transport-Security', value: 'max-age=63072000' },
  { key: 'Content-Security-Policy', value: "default-src 'self'; script-src 'self'
https://js.stripe.com; frame-ancestors 'none'; object-src 'none'; base-uri 'self'" },
]
```

How to confirm it's fixed: curl -sI https://example.com/ shows the Content-Security-Policy header.

A ready-to-paste fix prompt for your AI coding tool is in the fix pack under HG-EXAMPLE-0005.

HG-EXAMPLE-0006 · SERVER AND APP CONFIGURATION

Error page shows internal details

LOW | New | Confidence: confirmed | Effort: Minutes

What's wrong: When the contact page has a problem, the error message shows file names from the server.

Why it matters: It gives an attacker hints about how the site is built.

What to do: Show a plain error message and keep the details in the server logs.

Technical detail

Unhandled errors in the contact action render error.message to the user.

Impact: Information disclosure of paths and library names.

CVSS 4.0	2.3 `CVSS:4.0/AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N`
CWE	CWE-209
References	ASVS 5.0 V16.5.1, OWASP A10:2025

Where to look

- `app/contact/error.tsx:6`

Evidence

CODE

```
return <p>{error.message}</p>
```

How to fix it

Replace the message with plain text and log the error.

1. Render a fixed message.
2. Log error.digest server-side.

Exact change

`app/contact/error.tsx:6`: No internal details reach the visitor.

BEFORE

```
<p>{error.message}</p>
```

AFTER

```
<p>Something went wrong. Please try again, or email us.</p>
```

How to confirm it's fixed: Forcing an error on the contact page shows only the plain message.

A ready-to-paste fix prompt for your AI coding tool is in the fix pack under HG-EXAMPLE-0006.

Appendix

Technical summary

Full check (code review): Re-check of the main branch. Secrets in history resolved by rotation and a gitleaks CI job; jQuery removed. Object-level authorization is still missing on the booking detail handler. A new low finding: the contact error boundary renders error.message. DNS and email were out of scope this month.

Every check

Check	Result	Note
Every page and resource is served over HTTPS, with HTTP redirecting to HTTPS	Passed	
Certificate is valid, trusted, covers every hostname, and is not close to expiring	Passed	
Only TLS 1.2 and 1.3 are offered, with no weak cipher suites	Passed	
HSTS is set with a long max-age (and includeSubDomains/preload where safe)	Passed	
No mixed content (HTTP scripts, styles, images or form targets on HTTPS pages)	Passed	
Connections to databases, APIs and services inside the app are encrypted and verify certificates	Passed	
Content-Security-Policy is present and meaningful (no wildcard sources, unsafe-eval only if needed)	Problem found	
Clickjacking protection: frame-ancestors or X-Frame-Options	Passed	
X-Content-Type-Options: nosniff	Passed	
Referrer-Policy limits what leaks to other sites	Passed	
Permissions-Policy turns off camera, microphone, location and other features the site does not use	Passed	
Cross-Origin-Opener-Policy and Cross-Origin-Resource-Policy where appropriate	Passed	
CORS does not reflect any origin or allow credentials from untrusted origins	Passed	
Server and framework version headers (Server, X-Powered-By) are hidden	Passed	
Pages with personal data are not cached by shared caches	Passed	
JavaScript libraries with known vulnerabilities (jQuery, Bootstrap, Angular, Lodash and others) by version	Passed	
Third-party scripts: who they are, what they can reach, and whether they are needed	Passed	

Check	Result	Note
Scripts and styles from other domains use Subresource Integrity or come from a trusted pinned source	Passed	
Production source maps are not publicly downloadable	Passed	
No API keys, tokens or internal URLs in public JavaScript beyond those meant to be public	Passed	
DOM-based XSS: untrusted data written to innerHTML, eval, document.write, location or similar sinks	Passed	
postMessage handlers check the sender origin	Passed	
Tokens and personal data are not kept in localStorage where XSS could read them	Passed	
Redirects only go to allowed destinations	Passed	
Forms post over HTTPS to the expected domain, with autocomplete settings suited to the data	Passed	
robots.txt, sitemap.xml and security.txt: present where useful, and not pointing at private areas	Passed	
No directory listings on pages the site links to	Passed	
Error pages do not show stack traces, file paths, queries or framework debug output	Passed	
Page source has no developer comments with credentials, internal hosts or to-dos about security	Passed	
No backups, dumps, .env, .git, logs or config files deployed with the site	Passed	
Admin, staging and debug pages are not reachable by the public	Passed	
SPF record exists, lists only real senders, and ends in -all or ~all	Not checked	No live domain in scope for this check.
DMARC record exists with a policy of quarantine or reject, and reporting set up	Not checked	No live domain in scope for this check.
DKIM signing for the mail provider in use (common selectors checked)	Not checked	No live domain in scope for this check.
MTA-STS or TLS reporting for inbound mail where the domain receives email	Not checked	No live domain in scope for this check.
CAA record limits which authorities may issue certificates	Not checked	No live domain in scope for this check.
DNSSEC status noted	Not checked	No live domain in scope for this check.
No subdomains pointing at deleted services (dangling CNAMEs found through certificate transparency logs)	Not checked	No live domain in scope for this check.
Domain expiry date and registrar lock noted	Not checked	No live domain in scope for this check.

Check	Result	Note
CMS or framework version, and whether it still gets security updates	Passed	
Plugins, themes and extensions in use, with known vulnerabilities by version	Passed	
Default login and API paths that invite attacks (wp-login, xmlrpc.php, user enumeration through the REST API)	Passed	
Framework security settings: debug off, CSRF protection on, secure defaults kept	Passed	
Runtime and server software (Node, PHP, Python, database) still supported	Passed	
Packages with published vulnerabilities (direct and indirect), and whether the vulnerable code is actually used	Passed	
Lockfiles are committed and installs are reproducible	Passed	
Abandoned, deprecated or single-maintainer packages in sensitive places	Passed	
Package names that look like typos of popular ones, or install scripts that run unexpected code	Passed	
Docker images, GitHub Actions and CDNs pinned to versions or hashes	Passed	
Licenses noted where they create legal risk	Passed	
No secrets in source code, config files or tests	Passed	
No secrets anywhere in git history (still valid even if deleted later)	Passed	
Keys that are meant to be public (publishable keys, anon keys) cannot do more than intended	Passed	
Production secrets live in a secrets manager or environment settings, not shared files	Passed	
Each key has the smallest permissions it needs, and keys can be rotated	Passed	
Secrets are never written to logs, errors or analytics	Passed	
Password rules follow NIST 800-63B (length over complexity, breached-password check, no forced rotation)	Passed	
Passwords stored with a slow, salted hash (Argon2id, bcrypt or scrypt)	Passed	
Sign-in, sign-up and reset are rate limited and resist credential stuffing	Passed	
Sign-in, sign-up and reset do not reveal which emails have accounts	Passed	
Password reset tokens are single-use, expire quickly and are bound to the account	Passed	
MFA is available (and required for admins), with secure recovery	Passed	
Admin accounts are separate, protected and few	Passed	
No default, test or shared accounts in production	Passed	
Public sign-in pages use HTTPS and do not leak account existence in visible messages	Passed	

Check	Result	Note
Session cookies are Secure, HttpOnly, SameSite and use the __Host- prefix where possible	Passed	
Sessions rotate at sign-in, expire when idle and in total, and end everywhere at sign-out or password change	Passed	
State-changing requests are protected from cross-site request forgery	Passed	
JWTs: algorithm pinned (no "none"), signature checked, expiry short, audience and issuer checked, secrets strong	Passed	
OAuth and OIDC: PKCE, exact redirect URIs, state and nonce checked, tokens kept server-side	Passed	
API keys and personal access tokens can be listed, scoped and revoked by the user	Passed	
Every permission is checked on the server, never only hidden in the interface	Passed	
Users cannot reach other users' records by changing an ID (IDOR / broken object-level authorization)	Problem found	
Admin and staff functions reject ordinary users	Passed	
Multi-tenant data is separated per company or household in every query	Passed	
Users cannot set fields they should not (role, price, owner) through mass assignment	Passed	
Access is denied by default, and new routes are protected without extra work	Passed	
Exports, search, reports, webhooks and background jobs apply the same permissions as the main screens	Passed	
SQL and ORM raw queries use parameters, never string building	Passed	
NoSQL queries do not accept operator objects from users	Passed	
No shell commands built from user input	Passed	
Stored and reflected XSS: output is encoded for its context; raw HTML rendering is sanitized	Passed	
Server-side template injection	Passed	
XML parsers have external entities turned off (XXE)	Passed	
No header, log or email injection from user input (CRLF)	Passed	
No unsafe deserialization of user-supplied data (pickle, PHP unserialize, YAML load)	Passed	
LDAP, XPath and other query languages built safely	Passed	
Regular expressions on user input cannot hang the server (ReDoS)	Passed	
All input is validated on the server for type, length, range and format	Passed	
Multi-step flows cannot be skipped or replayed (checkout, onboarding, approvals)	Passed	

Check	Result	Note
Prices, discounts, quantities and totals are computed on the server	Passed	
Race conditions on money, credits, inventory, invites and one-time actions	Passed	
Business limits (trial length, seats, quotas) are enforced on the server	Passed	
Sensitive flows resist automated abuse (sign-up spam, coupon farming, scraping)	Passed	
Every API route is known, documented and protected; old versions are retired	Passed	
Responses only include fields the caller needs (no full user objects, hashes or internal flags)	Passed	
GraphQL: introspection off in production, depth and cost limits, field-level permissions	Passed	
Rate limits and pagination caps on every endpoint	Passed	
Strict content types and request size limits	Passed	
Publicly reachable API docs or GraphQL endpoints do not expose internal operations	Passed	
Uploads are checked by content, not just extension, with size limits	Passed	
Uploads are stored outside the web root or in private storage, served with safe content types	Passed	
File paths built from user input cannot escape their folder (../)	Passed	
Archives and images are processed safely (zip slip, decompression bombs, image library bugs)	Passed	
Storage buckets and file links are private unless meant to be public; signed links expire	Passed	
Features that fetch user-supplied URLs (previews, imports, webhooks, image proxies) block internal addresses	Passed	
Cloud metadata endpoints cannot be reached from app code paths	Passed	
Server-side fetches do not follow redirects to internal addresses	Passed	
No home-made or outdated cryptography (MD5, SHA-1 for security, ECB, static IVs)	Passed	
Tokens and IDs that must be secret use a secure random generator	Passed	
Sensitive data is encrypted at rest where it matters, with keys kept apart from the data	Passed	
Secrets and signatures are compared in constant time	Passed	
The privacy policy matches what the site actually collects, tracks and shares	Passed	
Tracking and analytics scripts, and whether they load before consent where consent is required	Passed	
Only the personal data needed is collected and kept	Passed	

Check	Result	Note
Data is deleted when promised; backups and logs follow the same rules	Passed	
Personal data is not written to logs, URLs, analytics or error trackers	Passed	
Users can export and delete their data where the law or policy says they can	Passed	
Debug mode, verbose errors and development tools are off in production	Problem found	
Staging and production are separated, with separate secrets and data	Passed	
Default credentials, sample apps and unused services are removed	Passed	
Only the HTTP methods each route needs are allowed	Passed	
Security settings are in code or config, reviewed, and the same in every environment	Passed	
Sign-ins, failures, permission denials, admin actions and data exports are logged	Passed	
Someone is alerted to repeated failures, unusual activity and errors	Passed	
Logs cannot be changed by the app's users and are kept long enough to investigate	Passed	
Errors and exceptions fail closed (deny access) rather than open	Passed	
Unhandled errors, timeouts and partial failures leave data consistent	Passed	
GitHub Actions do not run untrusted input in scripts (titles, branch names, pull_request_target)	Passed	
Workflow tokens have least privilege; secrets are not exposed to forks	Passed	
Main branch requires review; deploys come only from reviewed code	Passed	
Deploy keys and cloud credentials are scoped, short-lived where possible (OIDC), and rotated	Passed	
Third-party actions and build tools are pinned and trusted	Passed	
Infrastructure-as-code (Terraform, CloudFormation, Vercel/Netlify config) has no public resources by mistake	Passed	
Containers run as non-root from minimal, patched images	Passed	
Databases and internal services are not reachable from the internet	Passed	
Cloud accounts use least privilege, MFA and no long-lived root keys	Passed	
Backups exist, are encrypted, are tested, and cannot be deleted by the app	Passed	
CDN or firewall in front of the origin, and the origin cannot be reached directly	Passed	
Row-level security (Postgres/Supabase) is on for every exposed table, with correct policies	Passed	
Table and function grants to anon/authenticated roles are deliberate (Supabase public-schema grant changes)	Passed	
Firebase or Firestore security rules do not allow open reads or writes	Passed	

Check	Result	Note
Service-role or admin database keys never reach the browser	Passed	
The app connects with a database user that has only the rights it needs	Passed	
Prompt injection from users, documents, web pages or tool results cannot change what the AI is allowed to do	N/A	The site has no AI or real-time features.
AI tools act with the user's permissions only; important actions need a person to approve	N/A	The site has no AI or real-time features.
System prompts, other users' data and secrets cannot be pulled out through the AI	N/A	The site has no AI or real-time features.
AI output is treated as untrusted (no raw HTML, SQL or shell from model output)	N/A	The site has no AI or real-time features.
Per-user limits and spending caps stop abuse of paid AI calls	N/A	The site has no AI or real-time features.
A public chatbot does not reveal internal instructions or private data in normal use	N/A	The site has no AI or real-time features.
Card data never touches the app's servers (hosted fields or checkout); PCI scope noted	Passed	
Incoming webhooks verify signatures and reject replays	Passed	
Connected apps request the smallest scopes and tokens are stored safely	Passed	
Data from third-party APIs is validated before use	Passed	
Rate limits on forms, sign-in, search and expensive endpoints	Passed	
Spam and bot protection on public forms	Passed	
Slow or costly operations (exports, AI calls, image processing) are limited and queued	Passed	
Cloud spending alerts and caps exist	Passed	
WebSocket and real-time channels check who is connected and what they can subscribe to	N/A	The site has no AI or real-time features.
WebSocket connections check the Origin header	N/A	The site has no AI or real-time features.
WebRTC: TURN credentials are short-lived; media and data channels are encrypted	N/A	The site has no AI or real-time features.
Trust boundaries are clear: what is trusted, what is checked, and where	Passed	
Security controls are central and reused, not copied per route	Passed	
Each part of the system has only the access it needs	Passed	
Security-relevant behavior (permissions, validation) has automated tests	Passed	

Tools and model

Tool	Check	Status	Note
gitleaks 8.21.2	Full check (code review)	Ran	
osv-scanner 2.0.0	Full check (code review)	Ran	
semgrep 1.95.0	Full check (code review)	Ran	p/default, p/owasp-top-ten, p/typescript
trivy	Full check (code review)	Not run	Not installed; Dockerfile and Vercel config reviewed by hand.

Model: Claude Opus 5.5. Every finding in this report was reviewed by a person before it was included.

Severity

Severity uses CVSS version 4.0 base scores: Critical 9.0–10, High 7.0–8.9, Medium 4.0–6.9, Low 0.1–3.9. Info means an observation with no real risk. CWE is the industry's standard list of weakness types.

Terms of this report

What it covers Only the scope, dates, code version and checks shown in this report. It is a review at a point in time, not a promise that the site is secure or that every problem was found.

Your decisions You decide what to fix and how. Fix steps, code changes and AI prompts are suggestions: back up, review and test them before using them on a live site.

No further obligation This report doesn't commit HTMLGuy LLC to any further work, monitoring or re-checks. Other work happens only under a separate signed agreement, which controls where it differs from these terms.

Limit of liability To the fullest extent the law allows, HTMLGuy LLC's total liability for this report is limited to the amount paid for it (nothing, for a free quick check), and HTMLGuy LLC isn't liable for indirect or consequential losses, such as lost profits, data or business.

Who can rely on it This report is confidential and only for the client named on the cover. No one else may rely on it. It isn't legal or compliance advice.

Questions: https://htmlguy.com/contact