PCI DSS Web Report

Payment Card Industry Data Security Standard 4.0

PREPARED FOR	**Riverside Family Law, PLLC**
TARGET	**https://www.riversidefamilylaw.com**
SCAN TYPE	**Full Scan**
SCAN STARTED	**Apr 15, 2026, 9:42:18 AM PDT**
SCAN DURATION	**22 minutes, 18 seconds**
GENERATED	**Apr 15, 2026**
GENERATED BY	**RiskMeter Cybersecurity**

This is a sample document. Findings, target, and metadata are fabricated for demonstration purposes. Real RiskMeter reports describe actual scans of explicitly authorized assets.

SAMPLE · ILLUSTRATIVE

Description

The Payment Card Industry Data Security Standard (PCI DSS) was developed to encourage and enhance cardholder-data security and to facilitate the broad adoption of consistent data-security measures globally. PCI DSS provides a baseline of technical and operational requirements designed to protect cardholder data. It is intended for all entities that store, process, or transmit cardholder data (CHD) and/or sensitive authentication data (SAD) or could impact the security of the cardholder-data environment (CDE).

How RiskMeter maps to PCI DSS

A RiskMeter scan addresses the technical, web-application portion of PCI DSS — the requirements that test whether the public-facing web surface is configured to protect data in transit, prevent injection and session-management flaws, and avoid information disclosure. The scan **does not** evaluate the full PCI DSS scope: physical security, policies and procedures, internal-network segmentation, cardholder-data storage, key management, vendor management, and personnel training all require separate evaluation and are out of scope for this report.

Each requirement section below summarizes the requirement, describes the related good practice, and lists any web-surface findings that may indicate a gap. RiskMeter does not assert PCI compliance; this report is a technical input to a broader compliance program.

Disclaimer

This document is a technical scanning artifact. It does not constitute legal or compliance advice. Consult a Qualified Security Assessor (QSA) or your acquirer for an authoritative determination of PCI DSS compliance.

SAMPLE · ILLUSTRATIVE

Compliance at a glance

Number of findings observed under each PCI DSS requirement covered by this scan. Requirements not listed are out of scope for an external web scan.

Requirement	Findings
Requirement 2.2.6 — System security parameters configured to prevent misuse	3
Requirement 6.5.5 — Improper error handling and information disclosure	4
Requirement 6.2 — All system components are protected from known vulnerabilities by installing applicable security patches/updates	2
Requirement 8.3.4 — Invalid authentication attempts are limited	1
Requirement 4.2.1 — Strong cryptography and security protocols are used during transmission	3
Requirement 6.5.9 — Cross-site request forgery (CSRF) is addressed	1
Requirement 6.5.10 — Broken authentication and session management	2
Requirement 12.6 — Security awareness program	1

SAMPLE · ILLUSTRATIVE

Findings by requirement

Requirement 2.2.6 — System security parameters configured to prevent misuse

Purpose: Configure system security parameters to prevent misuse. **Good practice:** harden defaults; disable unnecessary services; remove vendor identifiers from public responses.

Backup file exposed (wp-config.php.bak)

Critical

CWE CWE-538 CVSS AV:N/AC:L/Au:N/C:C/I:N/A:N Instances 1

DESCRIPTION

A backup of the WordPress configuration file was discovered at a predictable URL. The file contains the database hostname, database user, database password, and WordPress secret keys in plaintext.

IMPACT

An attacker who downloads this file gains direct credentials to the site database, the ability to forge authentication cookies, and the ability to read or modify any data the WordPress process can access — including all client intake submissions stored in the site's database.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/wp-config.php.bak

```
GET /wp-config.php.bak HTTP/1.1
Host: www.riversidefamilylaw.com
Accept: */*
User-Agent: RiskMeter-Scanner/1.0

HTTP/1.1 200 OK
Content-Type: application/octet-stream
Content-Length: 4831

define( 'DB_NAME', 'rfl_wp_prod' );
define( 'DB_USER', 'rfl_wp_app' );
define( 'DB_PASSWORD', '[REDACTED]' );
...
```

RECOMMENDED FIX

Remove the backup file from the public web root immediately, then rotate every credential it contained.

Delete *wp-config.php.bak* and any sibling files matching **.bak*, **.old*, or **.orig* from the document root. Rotate the database password and the WordPress AUTH/SECURE/LOGGED_IN/NONCE keys (regenerate from the WordPress.org secret-key service). Audit the database for unexpected administrator accounts. Configure the web server to deny requests for files with backup extensions and prevent editor backups from being written into the web root.

REFERENCES

· OWASP Top 10 — A05:2021 Security Misconfiguration

· CWE-538: File and Directory Information Exposure

Server version disclosed in HTTP header

Informational

CWE CWE-200 CVSS AV:N/AC:L/Au:N/C:P/I:N/A:N Instances 1

DESCRIPTION

The Server response header reveals *Apache/2.4.41 (Ubuntu)*, exposing both the web server vendor and the precise patch level.

IMPACT

Attackers running automated tools can match a specific Apache patch level against the public CVE database to focus their exploitation attempts. This is reconnaissance, not exploitation, but every reduction in attacker certainty is a small win.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

```
HTTP/1.1 200 OK
Server: Apache/2.4.41 (Ubuntu)
```

RECOMMENDED FIX

Suppress the version string in the Server header.

Apache: set *ServerTokens Prod* and *ServerSignature Off*. Nginx: set *server_tokens off;*. Verify via curl that the response now shows *Server: Apache* with no version. Apply the same principle to PHP's X-Powered-By header (disable via *expose_php = Off*).

REFERENCES

· OWASP — Server-Side Information Leakage

· CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

WordPress version disclosed in generator meta tag

Informational

CWE CWE-200 CVSS AV:N/AC:L/Au:N/C:P/I:N/A:N Instances 1

DESCRIPTION

Pages include a tag, which exposes the precise WordPress version to anyone reading the HTML source.

IMPACT

Same reconnaissance principle as the server-version finding — the version makes it easy for attackers to match against known WordPress core CVEs.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

RECOMMENDED FIX

Remove the generator meta tag and the version query string from enqueued asset URLs.

Add a small mu-plugin or theme-functions snippet that calls *remove_action('wp_head', 'wp_generator')* and strips *?ver=* query strings from script and style enqueues. Combine with the WordPress core update (see the High-severity core finding) so the version is both hidden and current.

REFERENCES

· CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

SAMPLE · ILLUSTRATIVE

Requirement 6.5.5 — Improper error handling and information disclosure

Purpose: Prevent applications from leaking sensitive technical details that ease an attacker's reconnaissance.
Good practice: suppress version banners; deny direct access to dotfiles, backups, and source repositories; configure error pages.

.git repository exposed on web root

High

CWE CWE-540 CVSS AV:N/AC:L/Au:N/C:P/I:N/A:N Instances 1

DESCRIPTION

The /.git/ directory is accessible from the public internet. An attacker can reconstruct the entire source-code repository, including the full commit history, branches, and any files ever committed.

IMPACT

Source code disclosure exposes business logic, hard-coded credentials, internal hostnames, third-party API keys committed in earlier history, and any sensitive data that has ever been added and removed from the repository.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/.git/HEAD

```
GET /.git/HEAD HTTP/1.1
Host: www.riversidefamilylaw.com
User-Agent: RiskMeter-Scanner/1.0

HTTP/1.1 200 OK
Content-Type: text/plain

ref: refs/heads/main
```

RECOMMENDED FIX

Remove the .git directory from the web root and configure the server to deny access to dotfile paths.

Delete the /.git directory from the deployed site. Use a deploy process that copies built artifacts only — never the working repository. Add a server rule (Apache: *RedirectMatch 404 /\.git*; Nginx: *location ~ /\. { deny all; }*) so any future dotfile leaks are blocked at the edge. Audit the leaked history for any committed secrets and rotate them.

REFERENCES

· OWASP Top 10 — A05:2021 Security Misconfiguration

· CWE-540: Inclusion of Sensitive Information in Source Code

Directory listing enabled on /wp-content/uploads/

Medium

CWE CWE-548 CVSS AV:N/AC:L/Au:N/C:P/I:N/A:N Instances 1

DESCRIPTION

The /wp-content/uploads/ path returns a server-generated index listing of all uploaded files when accessed directly. Year/month subdirectories are also browsable.

IMPACT

Files that were intended to be sent only to a specific recipient (case PDFs, client photos, intake attachments) become discoverable by URL guessing or full directory enumeration. Older media uploaded years ago is just as discoverable as today's uploads.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/wp-content/uploads/

```
GET /wp-content/uploads/ HTTP/1.1

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

Index of /wp-content/uploads/...
2024/
2025/
2026/
```

RECOMMENDED FIX

Disable automatic directory listing on the web server and add a blank index file as a defense-in-depth measure.

Apache: set *Options -Indexes* in the relevant configuration block. Nginx: directory listing is off by default; verify *autoindex* is not enabled anywhere. Place an empty *index.html* in /wp-content/uploads/. Review what is currently in the uploads directory and move anything sensitive into a non-public storage location with access controls.

REFERENCES

· CWE-548: Exposure of Information Through Directory Listing

X-Frame-Options header missing (clickjacking)

Low

CWE CWE-1021 **CVSS** AV:N/AC:M/Au:N/C:N/I:P/A:N **Instances** 1

DESCRIPTION

Responses do not include an X-Frame-Options or Content-Security-Policy frame-ancestors directive. Pages can be embedded in an attacker-controlled .

IMPACT

An attacker page can frame the firm's login page or contact form and overlay invisible UI to trick visitors into clicking or typing. This is a classic clickjacking pattern.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/wp-login.php

```
GET /wp-login.php HTTP/1.1

HTTP/1.1 200 OK
(neither X-Frame-Options nor CSP frame-ancestors is set)
```

RECOMMENDED FIX

Add a Content-Security-Policy with frame-ancestors 'self' (or 'none' for sensitive pages).

Set the response header *Content-Security-Policy: frame-ancestors 'self'* on all pages. For pages that should never be framed (login, payment), use *'none'*. X-Frame-Options is the legacy equivalent and is still supported for older

browsers.

REFERENCES

· OWASP — Clickjacking Defense Cheat Sheet

· CWE-1021: Improper Restriction of Rendered UI Layers

Permissions-Policy header not implemented

Informational

CWE CWE-693 CVSS AV:N/AC:H/Au:N/C:P/I:N/A:N Instances 1

DESCRIPTION

The Permissions-Policy response header is not set. By default modern browsers allow embedded content to request access to geolocation, camera, microphone, and other powerful APIs.

IMPACT

If a third-party script (analytics, embedded video, advertising tag) is ever compromised, it could request sensitive permissions from the user under the firm's domain. Permissions-Policy is a preventive control.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

```
(no Permissions-Policy response header observed)
```

RECOMMENDED FIX

Set a restrictive Permissions-Policy header that disables powerful browser APIs by default.

Add the response header *Permissions-Policy: camera=(), microphone=(), geolocation=(), payment=()* on all pages. Add specific origins only to features the site actually needs. This belongs in the same configuration block as the CSP and X-Frame-Options changes.

REFERENCES

· Mozilla Web Docs — Permissions-Policy

· CWE-693: Protection Mechanism Failure

Requirement 6.2 — All system components are protected from known vulnerabilities by installing applicable security patches/updates

Purpose: Ensure software components are protected from known vulnerabilities through timely application of security patches. **Good practice:** maintain an inventory of components and a documented patch cadence; rely on automatic updates where possible.

WordPress core is outdated (5.8.x — multiple known CVEs)

High

CWE CWE-1104 CVSS AV:N/AC:L/Au:N/C:P/I:P/A:P Instances 1

DESCRIPTION

The site is running WordPress 5.8.6 (released Jun 2022). Multiple high-severity vulnerabilities have been patched in the WordPress core since this version, including authenticated stored XSS, data-exposure issues, and SQL injection in core query handling.

IMPACT

Unpatched core vulnerabilities allow opportunistic attackers using automated exploit kits to take over the site. Several of the issues fixed since 5.8.6 are pre-authentication or require only a low-privilege contributor role.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

```
GET / HTTP/1.1
Host: www.riversidefamilylaw.com

HTTP/1.1 200 OK
...
```

RECOMMENDED FIX

Update WordPress core to the current stable major release and enable automatic minor updates.

Apply the current WordPress release in a staging environment, verify plugin and theme compatibility, and roll forward to production. Enable *WP_AUTO_UPDATE_CORE* for minor releases. Document a quarterly review of core, theme, and plugin versions, and remove the version exposure from the generator meta tag (see the related Informational finding).

REFERENCES

· WordPress Release Cycle — https://wordpress.org/about/roadmap/

· CWE-1104: Use of Unmaintained Third-Party Components

Outdated jQuery 1.7.2 — multiple known XSS vulnerabilities

Medium

CWE CWE-1104 CVSS AV:N/AC:M/Au:N/C:P/I:P/A:N Instances 1

DESCRIPTION

The site loads jQuery 1.7.2 (released Mar 2012). This version contains multiple cross-site scripting issues fixed in 1.9.0+ (CVE-2012-6708) and 3.0.0+ (CVE-2015-9251).

IMPACT

Pages that use jQuery's vulnerable selector or HTML-parse paths with attacker-influenced strings can be coerced into executing script in a visitor's browser. For a law-firm site, this could be used to phish for client login credentials at adjacent portals.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/wp-includes/js/jquery/jquery.js

```
GET /wp-includes/js/jquery/jquery.js HTTP/1.1

...
/*! jQuery v1.7.2 jquery.com | jquery.org/license */
```

RECOMMENDED FIX

Upgrade jQuery to a current stable release (3.7+ as of report date) and audit any custom code that depends on legacy behavior.

Update WordPress core (which bundles a recent jQuery), or if a theme or plugin pins an old version, identify the source and either update or replace it. Run the site through the jQuery Migrate plugin during transition to surface API differences. Remove inline event handlers that rely on deprecated behaviors.

REFERENCES

· CVE-2015-9251 — jQuery selector cross-site scripting

· CWE-1104: Use of Unmaintained Third-Party Components

Requirement 8.3.4 — Invalid authentication attempts are limited

Purpose: Limit the rate at which authentication credentials can be tested against the system, frustrating brute-force and credential-stuffing attacks. **Good practice:** per-account and per-IP rate limits; account lockout with administrator unlock.

WordPress login page lacks rate limiting

High

CWE CWE-307 **CVSS** AV:N/AC:L/Au:N/C:P/I:P/A:N **Instances** 1

DESCRIPTION

The /wp-login.php endpoint accepts an unlimited number of authentication attempts from a single source IP without throttling, lockout, or CAPTCHA challenge.

IMPACT

Attackers can run credential-stuffing and password-spraying attacks against the firm's user accounts. Once any administrator or editor account is compromised, the site can be used to host phishing pages, redirect intake-form submissions, or modify content.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/wp-login.php

```
POST /wp-login.php HTTP/1.1
Host: www.riversidefamilylaw.com
Content-Type: application/x-www-form-urlencoded

log=admin&pwd;=Password123&wp-submit;=Log+In

(observed: 250 sequential failed attempts from a single IP completed within 90 seconds with no
rate-limit response)
```

RECOMMENDED FIX

Enforce per-IP and per-account rate limits on /wp-login.php and /xmlrpc.php; add multi-factor authentication for all editor and administrator accounts.

Install a brute-force protection plugin (e.g., Limit Login Attempts Reloaded, Wordfence) or enforce rate limiting at the edge (Cloudflare, Sucuri, web server module). Require MFA for all users with publish or admin capability. Disable or rate-limit /xmlrpc.php — it is a parallel authentication endpoint that frequently bypasses front-door protection.

REFERENCES

· OWASP — Authentication Cheat Sheet

· CWE-307: Improper Restriction of Excessive Authentication Attempts

Requirement 4.2.1 — Strong cryptography and security protocols are used during transmission

Purpose: Protect cardholder data (and authentication context) during transmission over open or public networks. **Good practice:** use only currently strong cryptographic protocols and cipher suites; validate certificates; monitor expiration.

TLS/SSL weak cipher suites enabled

Medium

CWE CWE-326 **CVSS** AV:N/AC:H/Au:N/C:P/I:P/A:N **Instances** 1

DESCRIPTION

The server negotiates TLS 1.0 and TLS 1.1, and offers cipher suites using RC4 and 3DES. Modern browsers no longer accept these by default, but tools that initiate connections programmatically (including non-browser clients used by some clients' IT teams) may still negotiate down to them.

IMPACT

Weak TLS configurations are flagged by every modern compliance regime (PCI DSS 4.0, HIPAA, FedRAMP). They also produce visible downgrade warnings in security questionnaires and insurance underwriting reviews.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

```
Negotiated TLS 1.0 with cipher TLS_RSA_WITH_RC4_128_SHA
Negotiated TLS 1.1 with cipher TLS_RSA_WITH_3DES_EDE_CBC_SHA
Server cipher list includes: RC4-MD5, RC4-SHA, DES-CBC3-SHA
```

RECOMMENDED FIX

Disable TLS 1.0 and TLS 1.1, and remove RC4 and 3DES cipher suites. Require TLS 1.2 or 1.3 only.

Update the web server TLS configuration to follow the Mozilla Intermediate or Modern profile. Remove all cipher suites whose name contains RC4, 3DES, or DES. Restart the web server and re-test with the SSL Labs server test. The result should be A or A+ with no protocol or cipher warnings.

REFERENCES

· Mozilla Server-Side TLS Recommendations

· CWE-326: Inadequate Encryption Strength

SSL certificate expires within 30 days

Medium

CWE CWE-298 **CVSS** AV:N/AC:L/Au:N/C:N/I:N/A:P **Instances** 1

DESCRIPTION

The TLS certificate for www.riversidefamilylaw.com expires on May 11, 2026 — 26 days from the scan date.

IMPACT

If the certificate expires before renewal, every visitor to the site will see a browser warning, and any automated client (CRM integrations, intake form callbacks, payment processors) will stop trusting the site. Trust restoration takes hours and erodes client confidence.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

```
Subject: CN=www.riversidefamilylaw.com
Issuer: Let's Encrypt R3
Valid From: Feb 11, 2026
Valid Until: May 11, 2026
```

RECOMMENDED FIX

Renew the certificate now. Configure automated renewal so this is not a recurring issue.

If using Let's Encrypt, run *certbot renew* and confirm the automatic-renewal cron/systemd timer is functioning. If using a commercial CA, place the renewal order, install the new certificate, and reload the web server. Confirm certificate expiry monitoring is in place — most modern host providers and uptime services will alert at 30, 14, and 7 days before expiry.

REFERENCES

· Let's Encrypt — Renewing certificates

HSTS header missing

Low

CWE	CWE-319	CVSS	AV:N/AC:H/Au:N/C:P/I:N/A:N	Instances	1

DESCRIPTION

The HTTP Strict-Transport-Security response header is not set on any response from the site.

IMPACT

Without HSTS, a browser visiting the site for the first time over plain HTTP can be redirected away from HTTPS by an active network attacker. HSTS instructs the browser to only ever use HTTPS for the domain, even if the user types *http://* or follows a stale link.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

```
GET / HTTP/1.1

HTTP/1.1 200 OK
(no Strict-Transport-Security header in response)
```

RECOMMENDED FIX

Set Strict-Transport-Security with a long max-age and includeSubDomains.

Add the response header *Strict-Transport-Security: max-age=63072000; includeSubDomains*. Roll out gradually if the domain has subdomains served over HTTP — start with a short max-age, verify nothing breaks, then increase. Once stable, consider HSTS preload submission.

REFERENCES

· Mozilla Web Security — HSTS

· CWE-319: Cleartext Transmission of Sensitive Information

SAMPLE · ILLUSTRATIVE

Requirement 6.5.9 — Cross-site request forgery (CSRF) is addressed

Purpose: Prevent attackers from causing authenticated users to submit unintended state-changing requests. **Good practice:** use anti-CSRF tokens or origin/referer validation on every state-changing form.

Contact form lacks CSRF protection

Medium

CWE CWE-352 **CVSS** AV:N/AC:M/Au:N/C:N/I:P/A:N **Instances** 1

DESCRIPTION

The public contact form at /contact/ accepts cross-origin POST submissions and does not validate a CSRF token, anti-forgery nonce, or origin/referer header.

IMPACT

Attackers can craft pages elsewhere on the internet that, when visited, silently submit the firm's contact form. This results in spam, fake intake records that pollute the firm's leads, and potential business-process disruption when the firm follows up on fabricated submissions.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/contact/

```
POST /contact/ HTTP/1.1
Host: www.riversidefamilylaw.com
Origin: https://attacker.example.com
Content-Type: application/x-www-form-urlencoded

name=Test&email;=test@example.com&message;=hello

HTTP/1.1 200 OK
(submission accepted; no CSRF challenge)
```

RECOMMENDED FIX

Add a CSRF token to the contact form and validate it on the server side before accepting submissions.

Use the form plugin's built-in CSRF / nonce mechanism (most WordPress form plugins support this). At minimum, validate the Origin or Referer header server-side and reject submissions from unexpected origins. Add basic rate limiting and a honeypot field as a defense in depth.

REFERENCES

· OWASP — Cross-Site Request Forgery (CSRF)

· CWE-352: Cross-Site Request Forgery

Requirement 6.5.10 — Broken authentication and session management

Purpose: Protect session identifiers and authentication state from disclosure or hijacking. **Good practice:** set Secure and HttpOnly on every session cookie; bind sessions to TLS.

Cookies not marked as Secure

Low

CWE CWE-614 **CVSS** AV:N/AC:M/Au:N/C:P/I:N/A:N **Instances** 1

DESCRIPTION

Several session and preference cookies (wordpress_logged_in_*, wp-settings-*) are set without the Secure flag.

IMPACT

Without the Secure flag, cookies can be transmitted over plain HTTP if any path on the domain is ever served unencrypted. This becomes meaningful when combined with HSTS being absent (see the HSTS finding) — a man-in-the-middle attacker on a coffee-shop network has a viable downgrade path.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/wp-admin/

```
Set-Cookie: wordpress_logged_in_abc=user1|...; path=/; HttpOnly
(Secure flag missing)
```

RECOMMENDED FIX

Add the Secure flag to every cookie set by the application.

WordPress: define *FORCE_SSL_ADMIN* in wp-config.php and ensure the site URL is configured as https://. For custom cookies, set the Secure attribute explicitly when calling *setcookie()* or the equivalent. Combine with HSTS so browsers refuse plain-HTTP requests altogether.

REFERENCES

· OWASP — Session Management Cheat Sheet

· CWE-614: Sensitive Cookie Without 'Secure' Attribute

Cookies not marked as HttpOnly

Low

CWE CWE-1004 **CVSS** AV:N/AC:M/Au:N/C:P/I:N/A:N **Instances** 1

DESCRIPTION

Two non-essential cookies (wp-settings-time-*, _ga_visitor_pref) are set without the HttpOnly flag, making them readable by client-side JavaScript.

IMPACT

If a cross-site scripting flaw is ever introduced anywhere on the domain, attacker-supplied script can read these cookies. While these specific cookies are not session tokens, defense in depth recommends marking every cookie HttpOnly unless JavaScript explicitly needs it.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/

```
Set-Cookie: wp-settings-time-1=...; path=/; expires=...
(HttpOnly flag missing)
```

RECOMMENDED FIX

Add HttpOnly to every cookie that does not require client-side JavaScript access.

Audit every *setcookie()* call and every cookie set via *Set-Cookie* response headers. Default to HttpOnly. The few cases where JavaScript truly needs to read a cookie should be evaluated individually and documented.

REFERENCES

· OWASP — HttpOnly

· CWE-1004: Sensitive Cookie Without 'HttpOnly' Flag

SAMPLE · ILLUSTRATIVE

Requirement 12.6 — Security awareness program

Purpose: Maintain awareness of security responsibilities through ongoing training. **Good practice:** staff training on phishing, wire-fraud, and social engineering — particularly for staff who handle client communications.

Email address disclosure in page source

Informational

CWE	CWE-200	CVSS	AV:N/AC:L/Au:N/C:N/I:N/A:N	Instances	1

DESCRIPTION

Multiple staff email addresses appear as plain mailto: links in the page source, including the managing partner's direct address.

IMPACT

Plain-text mailto: addresses are scraped by bulk-email harvesters and used as input for phishing campaigns. Targeting law-firm partners with phishing is a known attacker behavior, particularly around wire transfers and case-document handoffs.

EVIDENCE

Affected URL: https://www.riversidefamilylaw.com/team/

```
Email partner@riversidefamilylaw.com
```

RECOMMENDED FIX

Replace plain mailto: links with a contact form, or obfuscate addresses behind a JavaScript-rendered link.

The strongest defense is to route inbound contact through a form (which is also the natural place to enforce the CSRF and rate-limiting controls noted elsewhere). If direct addresses must remain visible, render them via JavaScript at page load time so they are not present in the static HTML scrape. Train staff to recognize wire-transfer and case-document phishing patterns.

REFERENCES

· CWE-200: Exposure of Sensitive Information to an Unauthorized Actor