Your Business
August 25, 2026 · 07:29 AM

SECURITY REPORT

4
/10
MEDIUM

SCAN TARGET

gaiaonline.com

Open ports detected: **0** · Public IP: **184.32.216.103**

A risk score of 4/10 (MEDIUM) indicates your business has security issues that need attention. The higher the score, the greater the chance of a breach, ransomware, or data theft.

EXECUTIVE SUMMARY

The security scan of gaiaonline.com found 15 medium and low severity issues. No critical vulnerabilities were detected, but 7 issues should be addressed this week to harden your security posture. Detailed fix instructions are included for each finding.

✓ No high-risk open ports were detected from the internet — your firewall appears to be blocking dangerous services.

TOP RECOMMENDATIONS

1. Outdated TLS Version Accepted (TLSv1.1): Disable TLS 1.
2. Missing HSTS Header: Add the HSTS header to your web server.
3. Missing Content-Security-Policy (CSP): Add a Content-Security-Policy header, but roll it out carefully — a misconfigured CSP can break legitimate scripts/styles on the site.

DETAILED FINDINGS

Outdated TLS Version Accepted (TLSv1.1)

Port SSL · TLS Version — MEDIUM

WHAT IT IS	Server accepts TLSv1.1 which is deprecated and insecure since 2020
BUSINESS RISK	Security scanners and compliance audits (PCI-DSS, SOC 2, cyber-insurance questionnaires, etc.) flag outdated TLS versions as a failing item, and major browsers are gradually moving toward blocking these connections entirely.
REAL-WORLD EXAMPLE	Example: A compliance auditor or cyber-insurance questionnaire runs an automated scan, flags the outdated TLS version as a failed control, and the business has to scramble to fix it before a policy renewal or contract can close.
HOW TO FIX	Disable TLS 1.0 and 1.1 in your web server config. For Nginx: add 'ssl_protocols TLSv1.2 TLSv1.3;' to your server block. For Apache: set 'SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1' in ssl.conf. Then restart the server and verify at https://www.ssllabs.com/ssltest/
URGENCY	**Fix within 1 week**
REFERENCE	CWE-327 — Use of a Broken or Risky Cryptographic Algorithm The encryption method in use is outdated or weak enough that attackers can break it with modern tools.

Missing HSTS Header

Port HTTPS · HSTS Header — MEDIUM

WHAT IT IS	Missing Strict-Transport-Security (HSTS) — browsers aren't forced to always use HTTPS, leaving visitors open to downgrade attacks
BUSINESS RISK	An attacker on the same network as a visitor (public wifi, a compromised router, etc.) can trick their browser into using the insecure version of your site and intercept what they type — raising the odds of stolen logins or payment details.
REAL-WORLD EXAMPLE	Example: An attacker on a shared network intercepts a visitor's first request (which defaults to HTTP) before the redirect happens, and silently serves them a fake version of the page instead of the real site.
HOW TO FIX	Add the HSTS header to your web server. Nginx: add 'add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;' in your server block. Apache: add 'Header always set Strict-Transport-Security "max-age=31536000"' in your config. Cloudflare: SSL/TLS > Edge Certificates > enable HTTP Strict Transport Security.
URGENCY	**Fix within 1 week**
REFERENCE	CWE-319 — Cleartext Transmission of Sensitive Information Sensitive data is sent over the network unencrypted, so anyone monitoring the connection can read it.

Missing Content-Security-Policy (CSP)

Port HTTPS · Content Security Policy

MEDIUM

WHAT IT IS	Missing Content-Security-Policy (CSP) — this site has no CSP defense-in-depth layer, so if any cross-site scripting weakness exists elsewhere it's easier to exploit; this finding on its own doesn't mean an XSS vulnerability exists on this site
BUSINESS RISK	CSP is a defense-in-depth browser control, not evidence of an active vulnerability by itself — but if an attacker ever manages to slip malicious script onto your site through some other weakness (e.g. a vulnerable plugin or a comment field), a good CSP is often what stops that script from running and stealing customer data such as login sessions or payment details. Without it, that second layer of protection isn't there.
REAL-WORLD EXAMPLE	Example: A vulnerable comment form or compromised ad widget lets an attacker inject a script tag; with a properly scoped CSP in place, the browser would refuse to run it — without CSP, that same injected script runs freely and can forward a visitor's session cookie to the attacker.
HOW TO FIX	Add a Content-Security-Policy header, but roll it out carefully — a misconfigured CSP can break legitimate scripts/styles on the site. Start by deploying it in Report-Only mode first, which reports violations without blocking anything: 'add_header Content-Security-Policy-Report-Only "default-src \'self\'; script-src \'self\'; object-src \'none\'; report-uri /csp-report";' Review the reports for a while to catch anything the policy would break, adjust the policy, then switch the header name to 'Content-Security-Policy' (without '-Report-Only') to start enforcing it. For WordPress or complex sites, use https://csp-evaluator.withgoogle.com to help build the policy.
URGENCY	**Fix within 1 week**
REFERENCE	CWE-693 — Protection Mechanism Failure A security safeguard that should be protecting the system is missing, disabled, or not strong enough.

Insecure Cookie Flags (gaia55_sid)

Port HTTPS · Cookie Security

MEDIUM

WHAT IT IS	Cookie 'gaia55_sid' is missing the Secure, SameSite flag(s). Observed attributes: path=/, domain=.gaiaonline.com, httponly.
BUSINESS RISK	This cookie's name suggests it may hold a session or authentication token, though its actual contents weren't inspected. If it does, a cookie missing these flags is easier to steal through cross-site scripting or to intercept over an unencrypted connection — and a stolen session cookie can let an attacker impersonate that logged-in user without ever needing their password.
REAL-WORLD EXAMPLE	Example: A visitor on public wifi has their session cookie intercepted because it wasn't marked Secure, or a malicious ad script reads it directly because it wasn't marked HttpOnly — either way, the attacker is now logged in as that user without ever seeing their password.
HOW TO FIX	Add the missing flag(s) when setting this cookie: Secure (only send over HTTPS), HttpOnly (block JavaScript access), SameSite=Lax or Strict (limit cross-site sending). Most frameworks expose this as a one-line config option — e.g. Flask: app.config['SESSION_COOKIE_SECURE']=True, SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE='Lax'.
URGENCY	**Fix within 1 week**
REFERENCE	CWE-614, CWE-1275 — Sensitive Cookie in HTTPS Session Without 'Secure' Attribute; Sensitive Cookie with Improper SameSite Attribute The cookie isn't marked Secure, so a browser could send it over an unencrypted connection where it can be intercepted. The cookie's SameSite attribute isn't set to a safe value, making it easier for other sites to trick a visitor's browser into sending it in cross-site requests.

Apache Server Status Exposed (Unconfirmed)

Port HTTPS · Apache Server Status Exposed

MEDIUM

WHAT IT IS	The path /server-status returns a restricted-access response (401/403) that is genuinely distinct from how this site handles random nonexistent paths — this confirms *something* is being specially handled at this path, but the response contains no content confirming what it actually is — this exposes internal server diagnostic information, not a login panel
BUSINESS RISK	This page reveals internal details about the server's configuration, active connections, or recent errors — not something the general public should be able to see, and useful reconnaissance an attacker can use to plan a more targeted attack elsewhere on the site.
REAL-WORLD EXAMPLE	Example: An attacker reviews this page to learn internal server details, request patterns, or recent error messages, then uses that information to plan a more targeted attack elsewhere on the site instead of guessing blind.
HOW TO FIX	Restrict access to /server-status by IP allowlist, or disable it entirely if not needed. Nginx: 'location ^~ /server-status { allow YOUR_IP; deny all; }'. Apache ('server-status'/'server-info'): add 'Require ip YOUR_IP' inside the relevant <Location> block, or remove the module if unused.
URGENCY	**Fix within 1 week**
REFERENCE	CWE-200 — Exposure of Sensitive Information The system reveals information to someone who shouldn't have access to it.

Weak DKIM Key — 1024-bit RSA (selector: dkim)

Port DNS · DKIM Key Strength

MEDIUM

WHAT IT IS	DKIM key for selector 'dkim' appears to be approximately 1024 bits — 1024-bit RSA keys are considered weak by modern standards and NIST has deprecated them. A well-resourced attacker could factor this key, breaking your email authentication.
BUSINESS RISK	A forged DKIM signature lets an attacker send phishing or fraud emails that cryptographically appear to come from your domain — bypassing email filters that rely on DKIM as a trust signal, and making impersonation emails indistinguishable from your real ones.
REAL-WORLD EXAMPLE	Example: A security researcher demonstrated in a published study that 512-bit DKIM keys could be factored in under 72 hours using cloud computing resources costing less than $100 — allowing anyone with that capability to forge valid email signatures for the affected domain.
HOW TO FIX	Upgrade the DKIM key for selector 'dkim' to 2048 bits. Generate a new key through your email provider, update the DNS TXT record to the new public key, and retire the old 1024-bit key.
URGENCY	**Fix within 1 week**
REFERENCE	CWE-326

Third-Party Scripts Missing Subresource Integrity (SRI)

Port HTTPS · Third-Party Script Loading

MEDIUM

WHAT IT IS	5 third-party script source(s) loaded without Subresource Integrity (SRI): a.pub.network, cdn1.gaiaonline.com, graphics.gaiaonline.com, sm1.selectmedia.asia, www.googletagmanager.com
BUSINESS RISK	If any of these third-party providers is ever compromised — a real, recurring attack pattern called a 'watering hole' or supply-chain attack, where attackers hit a widely-trusted vendor instead of you directly — the malicious code they inject would run on your site with no verification and no warning to you or your visitors.
REAL-WORLD EXAMPLE	Example: A widely-used analytics or widget provider gets compromised (a real, recurring attack pattern), and because the script loads without integrity verification, the malicious version executes on every visitor's browser with no warning to you or them.
HOW TO FIX	Add integrity and crossorigin attributes to each third-party <script> tag, e.g. <script src="..." integrity="sha384-..." crossorigin="anonymous"></script>. Most CDNs (cdnjs, jsdelivr, unpkg) publish the correct hash on their site — copy it directly.
URGENCY	**Fix within 1 week**
REFERENCE	CWE-353 — Missing Support for Integrity Check There's no way to verify that data wasn't altered in transit, so tampering would go unnoticed.

Insecure Cookie Flags (gaia55_tag)

Port HTTPS · Cookie Security

LOW

WHAT IT IS	Cookie 'gaia55_tag' is missing the Secure, SameSite flag(s). Observed attributes: expires=Wed, 25-Aug-2027 07:28:50 GMT, path=/, domain=.gaiaonline.com, httponly.
BUSINESS RISK	This cookie's name doesn't clearly indicate it holds session or authentication data, so the practical impact depends on what value it actually stores — anywhere from low-stakes (a UI preference) to more sensitive (tracking or personalization data). Missing these flags means whatever the cookie does hold is more exposed than it needs to be to interception or script access.
REAL-WORLD EXAMPLE	Example: whatever value 'gaia55_tag' holds could be read by an injected script (no HttpOnly) or intercepted on an unencrypted connection (no Secure) — the actual severity depends on how sensitive that value turns out to be.
HOW TO FIX	Add the missing flag(s) when setting this cookie: Secure (only send over HTTPS), HttpOnly (block JavaScript access), SameSite=Lax or Strict (limit cross-site sending). Most frameworks expose this as a one-line config option — e.g. Flask: app.config['SESSION_COOKIE_SECURE']=True, SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE='Lax'.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-614, CWE-1275 — Sensitive Cookie in HTTPS Session Without 'Secure' Attribute; Sensitive Cookie with Improper SameSite Attribute The cookie isn't marked Secure, so a browser could send it over an unencrypted connection where it can be intercepted. The cookie's SameSite attribute isn't set to a safe value, making it easier for other sites to trick a visitor's browser into sending it in cross-site requests.

Insecure Cookie Flags (gaia55_prp)

Port HTTPS · Cookie Security

LOW

WHAT IT IS	Cookie 'gaia55_prp' is missing the Secure, SameSite flag(s). Observed attributes: expires=Thu, 24-Sep-2026 07:28:50 GMT, path=/, domain=.gaiaonline.com, httponly.
BUSINESS RISK	This cookie's name doesn't clearly indicate it holds session or authentication data, so the practical impact depends on what value it actually stores — anywhere from low-stakes (a UI preference) to more sensitive (tracking or personalization data). Missing these flags means whatever the cookie does hold is more exposed than it needs to be to interception or script access.
REAL-WORLD EXAMPLE	Example: whatever value 'gaia55_prp' holds could be read by an injected script (no HttpOnly) or intercepted on an unencrypted connection (no Secure) — the actual severity depends on how sensitive that value turns out to be.
HOW TO FIX	Add the missing flag(s) when setting this cookie: Secure (only send over HTTPS), HttpOnly (block JavaScript access), SameSite=Lax or Strict (limit cross-site sending). Most frameworks expose this as a one-line config option — e.g. Flask: app.config['SESSION_COOKIE_SECURE']=True, SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE='Lax'.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-614, CWE-1275 — Sensitive Cookie in HTTPS Session Without 'Secure' Attribute; Sensitive Cookie with Improper SameSite Attribute The cookie isn't marked Secure, so a browser could send it over an unencrypted connection where it can be intercepted. The cookie's SameSite attribute isn't set to a safe value, making it easier for other sites to trick a visitor's browser into sending it in cross-site requests.

Insecure Cookie Flags (gaia55_ano)

Port HTTPS · Cookie Security

LOW

WHAT IT IS	Cookie 'gaia55_ano' is missing the Secure, SameSite flag(s). Observed attributes: path=/, domain=.gaiaonline.com, httponly.
BUSINESS RISK	This cookie's name doesn't clearly indicate it holds session or authentication data, so the practical impact depends on what value it actually stores — anywhere from low-stakes (a UI preference) to more sensitive (tracking or personalization data). Missing these flags means whatever the cookie does hold is more exposed than it needs to be to interception or script access.
REAL-WORLD EXAMPLE	Example: whatever value 'gaia55_ano' holds could be read by an injected script (no HttpOnly) or intercepted on an unencrypted connection (no Secure) — the actual severity depends on how sensitive that value turns out to be.
HOW TO FIX	Add the missing flag(s) when setting this cookie: Secure (only send over HTTPS), HttpOnly (block JavaScript access), SameSite=Lax or Strict (limit cross-site sending). Most frameworks expose this as a one-line config option — e.g. Flask: app.config['SESSION_COOKIE_SECURE']=True, SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE='Lax'.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-614, CWE-1275 — Sensitive Cookie in HTTPS Session Without 'Secure' Attribute; Sensitive Cookie with Improper SameSite Attribute The cookie isn't marked Secure, so a browser could send it over an unencrypted connection where it can be intercepted. The cookie's SameSite attribute isn't set to a safe value, making it easier for other sites to trick a visitor's browser into sending it in cross-site requests.

Insecure Cookie Flags (hdr_town_name)

Port HTTPS · Cookie Security

LOW

WHAT IT IS	Cookie 'hdr_town_name' is missing the Secure, SameSite flag(s). Observed attributes: Domain=.gaiaonline.com, Path=/, HttpOnly.
BUSINESS RISK	This cookie's name doesn't clearly indicate it holds session or authentication data, so the practical impact depends on what value it actually stores — anywhere from low-stakes (a UI preference) to more sensitive (tracking or personalization data). Missing these flags means whatever the cookie does hold is more exposed than it needs to be to interception or script access.
REAL-WORLD EXAMPLE	Example: whatever value 'hdr_town_name' holds could be read by an injected script (no HttpOnly) or intercepted on an unencrypted connection (no Secure) — the actual severity depends on how sensitive that value turns out to be.
HOW TO FIX	Add the missing flag(s) when setting this cookie: Secure (only send over HTTPS), HttpOnly (block JavaScript access), SameSite=Lax or Strict (limit cross-site sending). Most frameworks expose this as a one-line config option — e.g. Flask: app.config['SESSION_COOKIE_SECURE']=True, SESSION_COOKIE_HTTPONLY=True, SESSION_COOKIE_SAMESITE='Lax'.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-614, CWE-1275 — Sensitive Cookie in HTTPS Session Without 'Secure' Attribute; Sensitive Cookie with Improper SameSite Attribute The cookie isn't marked Secure, so a browser could send it over an unencrypted connection where it can be intercepted. The cookie's SameSite attribute isn't set to a safe value, making it easier for other sites to trick a visitor's browser into sending it in cross-site requests.

Apache Server Info Exposed (Unconfirmed)

Port HTTPS · Apache Server Info Exposed

LOW

WHAT IT IS	The path /server-info returns a restricted-access response (401/403) that is genuinely distinct from how this site handles random nonexistent paths — this confirms *something* is being specially handled at this path, but the response contains no content confirming what it actually is — this exposes internal server diagnostic information, not a login panel
BUSINESS RISK	This page reveals internal details about the server's configuration, active connections, or recent errors — not something the general public should be able to see, and useful reconnaissance an attacker can use to plan a more targeted attack elsewhere on the site.
REAL-WORLD EXAMPLE	Example: An attacker reviews this page to learn internal server details, request patterns, or recent error messages, then uses that information to plan a more targeted attack elsewhere on the site instead of guessing blind.
HOW TO FIX	Restrict access to /server-info by IP allowlist, or disable it entirely if not needed. Nginx: 'location ^~ /server-info { allow YOUR_IP; deny all; }'. Apache ('server-status'/'server-info'): add 'Require ip YOUR_IP' inside the relevant <Location> block, or remove the module if unused.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-200 — Exposure of Sensitive Information The system reveals information to someone who shouldn't have access to it.

Protected Path — /.htpasswd

Port PATH · Content Discovery

LOW

WHAT IT IS	/.htpasswd returned HTTP 403. This confirms the path exists, but access control prevented access — it was not accessed.
BUSINESS RISK	/.htpasswd returned HTTP 403. This confirms the path exists, but access control prevented access — it was not accessed.
HOW TO FIX	No action needed if this is intentionally restricted. Confirm the credentials protecting it are strong and not left at a default.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-284

Protected Path — /server-status

Port PATH · Content Discovery — LOW

WHAT IT IS	/server-status returned HTTP 403. This confirms the path exists, but access control prevented access — it was not accessed.
BUSINESS RISK	/server-status returned HTTP 403. This confirms the path exists, but access control prevented access — it was not accessed.
HOW TO FIX	No action needed if this is intentionally restricted. Confirm the credentials protecting it are strong and not left at a default.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-284

Login/Admin Path Discovered — /admin/

Port PATH · Content Discovery — LOW

WHAT IT IS	/admin/ returned HTTP 200 with no authentication challenge. This confirms a login or admin entry point exists at this path.
BUSINESS RISK	/admin/ returned HTTP 200 with no authentication challenge. This confirms a login or admin entry point exists at this path.
HOW TO FIX	Confirm this login is protected by a strong password and, ideally, multi-factor authentication. Consider IP-restricting it if it's only used by internal staff.
URGENCY	**Fix within 1 month**
REFERENCE	CWE-284

PORT REFERENCE TABLE

PORT	SERVICE	RISK	DESCRIPTION
SSL	TLS Version	**MEDIUM**	Server accepts TLSv1.1 which is deprecated and insecure since 2020
HTTPS	HSTS Header	**MEDIUM**	Missing Strict-Transport-Security (HSTS) — browsers aren't forced to always use HTTPS, leaving visit
HTTPS	Content Security Policy	**MEDIUM**	Missing Content-Security-Policy (CSP) — this site has no CSP defense-in-depth layer, so if any cross
HTTPS	Cookie Security	**MEDIUM**	Cookie 'gaia55_sid' is missing the Secure, SameSite flag(s). Observed attributes: path=/, domain=.ga

HTTPS	Apache Server Status Exposed	**MEDIUM**	The path /server-status returns a restricted-access response (401/403) that is genuinely distinct fr
DNS	DKIM Key Strength	**MEDIUM**	DKIM key for selector 'dkim' appears to be approximately 1024 bits — 1024-bit RSA keys are considere
HTTPS	Third-Party Script Loading	**MEDIUM**	5 third-party script source(s) loaded without Subresource Integrity (SRI): a.pub.network, cdn1.gaiao
HTTPS	Cookie Security	**LOW**	Cookie 'gaia55_tag' is missing the Secure, SameSite flag(s). Observed attributes: expires=Wed, 25-Au
HTTPS	Cookie Security	**LOW**	Cookie 'gaia55_prp' is missing the Secure, SameSite flag(s). Observed attributes: expires=Thu, 24-Se
HTTPS	Cookie Security	**LOW**	Cookie 'gaia55_ano' is missing the Secure, SameSite flag(s). Observed attributes: path=/, domain=.ga
HTTPS	Cookie Security	**LOW**	Cookie 'hdr_town_name' is missing the Secure, SameSite flag(s). Observed attributes: Domain=.gaiaonl
HTTPS	Apache Server Info Exposed	**LOW**	The path /server-info returns a restricted-access response (401/403) that is genuinely distinct from
PATH	Content Discovery	**LOW**	/.htpasswd returned HTTP 403. This confirms the path exists, but access control prevented access — i
PATH	Content Discovery	**LOW**	/server-status returned HTTP 403. This confirms the path exists, but access control prevented access
PATH	Content Discovery	**LOW**	/admin/ returned HTTP 200 with no authentication challenge. This confirms a login or admin entry poi

NEXT STEPS

→ Address the 7 MEDIUM risk finding(s) this week. Start with 'Outdated TLS Version Accepted (TLSv1.1)': Disable TLS 1.