CyberTestify

Automated Security Scan Report

testasp.vulnweb.com · Basic Scan

CyberTestify Security Scan — Completed

Report No: **CT-ÖRNEK-0D8B**

Verification Code: **F1DE-4625**

This report is not a formal penetration test / certification.

Table of Contents

TARGET	PACKAGE	REPORT NO
testasp.vulnweb.com	Basic Scan	CT-ÖRNEK-0D8B

1. Executive Summary

OVERALL ASSESSMENT — High Risk

An urgent issue that can show visitors a direct security warning (e.g. certificate validity/hostname) was detected; it should be fixed with priority.

⚡ Management Decision

1 high/critical indicator(s) requiring priority attention were found; an urgent remediation plan is recommended. Medium/low items can be addressed via planned improvement.

- **Overall risk level: High** — the site does not support HTTPS; communication is carried unencrypted (plaintext) — it can be intercepted/modified. Migrating to HTTPS should be the priority.
- ⚠ This target did not respond over HTTPS (443); the scan was performed **over http://**. The lack of HTTPS is a finding on its own (below).
- 6/6 important security headers are missing: Content-Security-Policy, X-Frame-Options, X-Content-Type-Options, Strict-Transport-Security, Referrer-Policy, Permissions-Policy.
- **Scope:** The security-header and version-signature checks were run on **8 unique pages** including the home page (not just a single page).
- **Recommended first step:** Install a valid TLS certificate and move all traffic to HTTPS; ready-made step-by-step commands are provided in the "AI Solution Recommendations" add-on.

Scope and limits: This package is a **passive, GET-based** external observation; no active exploitation or probe was attempted. A "no finding" statement in an area **does NOT prove** it is secure, because no active test was performed — it only shows that the externally observed configuration is clean.

Improvement Priorities

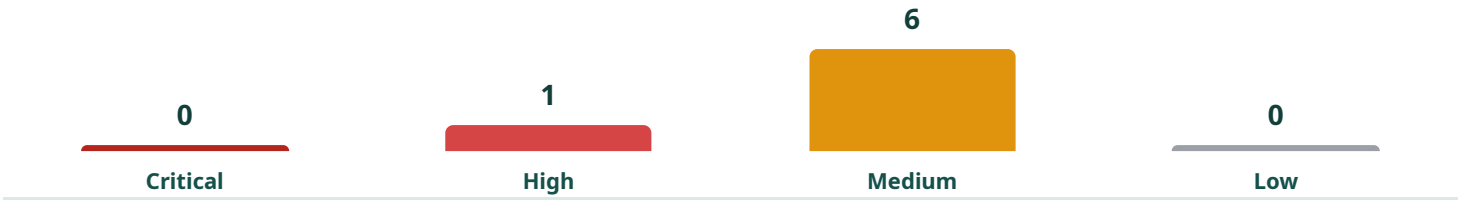
⚡ **Most Urgent:** HTTPS not supported (unencrypted communication).

🛠 **Quick Wins (~1-2 days):** Missing Content-Security-Policy; Missing X-Frame-Options (clickjacking); Missing X-Content-Type-Options; Missing HSTS; Missing Referrer-Policy.

🗓 **Medium-term / Process:** Missing Permissions-Policy.

2. Findings

2.1 Vulnerability Distribution



A total of **7** finding indicators were detected; the severity distribution is below.

2.2 Master Findings Table

ID	Title	State	Severity
CT-1	HTTPS not supported (unencrypted communication)	Open	High
CT-2	Missing Content-Security-Policy	Open	Medium
CT-3	Missing X-Frame-Options (clickjacking)	Open	Medium
CT-4	Missing X-Content-Type-Options	Open	Medium
CT-5	Missing HSTS	Open	Medium
CT-6	Missing Referrer-Policy	Open	Medium
CT-7	Missing Permissions-Policy	Open	Medium

3 control areas were run — 1 came back clean, 2 produced a finding

✓ CHECKED — NO ISSUE FOUND

- **Server/software version signature (on 8 pages)** No issue found (no known old/EOL version signature detected)

⚠ FINDING PRESENT — DETAILED IN THE SECTIONS BELOW

- **HTTP security headers (on 8 pages)** Finding present (6/6 recommended headers missing — detailed above)
- **TLS / certificate** Finding present (HTTPS did not respond — unencrypted communication)

2.3 Detailed Findings

[MEDIUM] **CT-2 · Missing Content-Security-Policy**

State: Open

Description: No Content-Security-Policy; no browser-side XSS mitigation.

How it was detected: The browser cannot restrict which resources are loaded; there is no basic defence against XSS and content injection. Missing on ALL 8 scanned unique pages.

Business Impact: No browser-side mitigation against content injection/XSS; injected scripts can execute.

RECOMMENDED FIX

Define a strict CSP (`default-src 'self'`) and allowlist third-party sources.

Reference: CWE-693 · OWASP A05:2021 Security Misconfiguration

[MEDIUM] CT-3 · Missing X-Frame-Options (clickjacking)

State: Open

Description: The page can be framed by other sites (no X-Frame-Options / CSP frame-ancestors).

How it was detected: The page can be embedded in another site's iframe; users can be deceived via clickjacking. Missing on ALL 8 scanned unique pages.

Business Impact: The page can be framed invisibly to trick users into unintended actions (clickjacking), weakening account and transaction safety.

RECOMMENDED FIX

Add `X-Frame-Options: SAMEORIGIN` or CSP `frame-ancestors 'self'`.

Reference: CWE-1021 · OWASP A05:2021 Security Misconfiguration

[MEDIUM] CT-4 · Missing X-Content-Type-Options

State: Open

Description: No X-Content-Type-Options; browser may MIME-sniff.

How it was detected: The browser may guess the content type (MIME-sniffing); uploaded files could be executed as scripts. Missing on ALL 8 scanned unique pages.

Business Impact: The browser may sniff content types and execute malicious content, widening the XSS/injection surface.

RECOMMENDED FIX

Add `X-Content-Type-Options: nosniff`.

Reference: CWE-693 · OWASP A05:2021 Security Misconfiguration

[MEDIUM] CT-5 · Missing HSTS

State: Open

Description: No HSTS; browser not forced to HTTPS.

How it was detected: HTTPS is not enforced to the browser; on first requests there is an SSL-stripping/MITM risk. Missing on ALL 8 scanned unique pages.

Business Impact: Without forced HTTPS, traffic can be intercepted/redirected via man-in-the-middle.

RECOMMENDED FIX

Add `Strict-Transport-Security: max-age=31536000; includeSubDomains` (if fully HTTPS).

Reference: CWE-319 · OWASP A05:2021 Security Misconfiguration

[MEDIUM] CT-6 · Missing Referrer-Policy

State: Open

Description: No Referrer-Policy; full URL may leak to external links.

How it was detected: The full URL (Referer) is sent to external links; session/privacy information may leak. Missing on ALL 8 scanned unique pages.

Business Impact: Address/session info may leak to external links; low-impact information exposure.

RECOMMENDED FIX

Add `Referrer-Policy: strict-origin-when-cross-origin`.

Reference: CWE-200 · OWASP A05:2021 Security Misconfiguration

[MEDIUM] **CT-7 · Missing Permissions-Policy**

State: Open

Description: No Permissions-Policy header; browser feature access is unrestricted.

How it was detected: Sensitive APIs such as camera/microphone/location are not restricted; third-party content could abuse them. Missing on ALL 8 scanned unique pages.

Business Impact: No Permissions-Policy; browser feature access (camera/mic/geolocation) is unrestricted (informational hardening).

RECOMMENDED FIX

Add a `Permissions-Policy` disabling unused features (e.g. `geolocation=(), camera=(), microphone=()`).

Reference: CWE-693 · OWASP A05:2021 Security Misconfiguration

3. Controls & Methodology

HTTP SECURITY HEADERS

Header	Status	Description
Strict-Transport-Security	Missing	The browser is not told to enforce HTTPS; there is an SSL-stripping/MITM risk on first requests.
Content-Security-Policy	Missing	The browser cannot restrict which resources are loaded; there is no basic defence against XSS and content injection.
X-Frame-Options	Missing	The page can be embedded in another site's iframe; the user can be tricked via clickjacking.
X-Content-Type-Options	Missing	The browser may guess the content type (MIME-sniffing); uploaded files could run as scripts.
Referrer-Policy	Missing	The full URL (Referer) is sent to external links; session/privacy information can leak.
Permissions-Policy	Missing	Sensitive APIs such as camera/microphone/location are not restricted; third-party content could abuse them.
X-XSS-Protection	Missing	The legacy-browser XSS filter is not set (not critical in modern browsers; the real protection is the CSP).
Content-Type	Present	text/html

TLS CERTIFICATE STATUS

⚠ This target **did not respond over HTTPS (443)**; no valid TLS certificate was found. The site is live only over **unencrypted HTTP** (see Identified Risks → "HTTPS not supported"). The header checks below were performed over http://.

SERVER / TECHNOLOGY SIGNATURE

- Server: Microsoft-IIS/8.5
- X-Powered-By: ASP.NET

IDENTIFIED RISKS

Finding	Severity	Description
HTTPS not supported (unencrypted communication)	High	The site does not respond over HTTPS; all traffic to/from the page is carried unencrypted (plaintext) — an attacker on the same network can intercept the traffic, steal sessions/passwords or modify content. The scan was performed over http://.
Critical security header missing: Content-Security-Policy	Medium	The browser cannot restrict which resources are loaded; there is no basic defence against XSS and content injection. Missing on ALL 8 scanned unique pages.
Critical security header missing: X-Frame-Options	Medium	The page can be embedded in another site's iframe; users can be deceived via clickjacking. Missing on ALL 8 scanned unique pages.
Security header missing: X-Content-Type-Options	Medium	The browser may guess the content type (MIME-sniffing); uploaded files could be executed as scripts. Missing on ALL 8 scanned unique pages.
Security header missing: Strict-Transport-Security	Medium	HTTPS is not enforced to the browser; on first requests there is an SSL-stripping/MITM risk. Missing on ALL 8 scanned unique pages.
Security header missing: Referrer-Policy	Medium	The full URL (Referer) is sent to external links; session/privacy information may leak. Missing on ALL 8 scanned unique pages.
Security header missing: Permissions-Policy	Medium	Sensitive APIs such as camera/microphone/location are not restricted; third-party content could abuse them. Missing on ALL 8 scanned unique pages.

POSITIVE ASSURANCE — AREAS CHECKED

Including the areas that produced no finding, the Basic Scan checks were actually run on **8 unique pages** including the home page. The table below also shows the "no issue found" results transparently:

Control Area	Result
HTTP security headers (on 8 pages)	⚠ **Finding present (6/6 recommended headers missing — detailed above)**
TLS / certificate	⚠ **Finding present (HTTPS did not respond — unencrypted communication)**
Server/software version signature (on 8 pages)	✓ **No issue found (no known old/EOL version signature detected)**

> **Three-state distinction (honesty):** ✓ *No issue found* = the check ran, came back clean · ⚠ *Finding present* = detailed above · ⚠ *Not assessable* = no data collected (does NOT mean secure).

What this package DOES and does NOT check

> **DOES (passive — page fetch via GET only, no probe/payload is sent):** HTTP security headers, TLS/certificate status (validity · hostname · TLS version), server-software version signature and known old/EOL version detection — across the 8 discovered pages.

> **DOES NOT:** CORS policy, cookie flag detail, Content-Security-Policy analysis, DNS/e-mail records (SPF/DKIM/DMARC) and exposed sensitive-file scanning are in the **External Surface** package; GDPR/PCI/ISO framework mapping is in the **Compliance** package; subdomain/API/CVE discovery is in the **Reconnaissance** package; active vulnerability verification

(SQLi/XSS/IDOR probing) is addressed in the **Active Verification** and **Full-Scope Pentest** packages. This report is based on passive observation; "no finding" **does NOT prove** it is secure, because no active exploitation was attempted.

Methodology

Stage	Description
Discovery	The external surface, pages and (for SPAs) real endpoints/params from the JS bundle are extracted.
Automated detection	Deterministic, safe (read-only) indicators are sought on the discovered surface.
Manual-deterministic verification	Findings are verified with code-based rules; “prove, don't exploit”.
Authz & session	In the authenticated package, cookie/session/authorization and post-login surface are examined.
Configuration	Header, TLS, email/DNS and exposure configurations are observed.

Risk Rating Criteria

Severity	Definition
Critical	Directly/easily exploitable indicator with serious data/access impact.
High	Prominent, priority-to-fix strong indicator.
Medium	Requires attention; context-dependent verification recommended.
Low	Hardening/maturity opportunity; low priority.

Severity is a band label only (no numeric CVSS score); all findings are framed as “indicator, verification required”.

---

4. AI Fix Suggestions

The recommendations below aim to fix the missing security headers detected on the testasp.vulnweb.com home page. For each header, a short explanation is given first, followed by an Nginx example; at the end you will find ready-made blocks for non-Nginx platforms (Firebase, Vercel, Next.js, Apache). Copy the one that fits your server.

1. Add Content-Security-Policy (CSP)

This is the most effective browser defence against XSS and content injection. Start with a basic policy suited to your site and tighten it over time:

```
add_header Content-Security-Policy "default-src 'self'; img-src 'self' data:; script-src 'self'; style-src '
```

If you use third-party scripts (GTM, Analytics, Pixel), add the relevant domains to `script-src` / `connect-src`.

2. Add X-Frame-Options

Prevents your page from being embedded in another site's iframe and exposed to clickjacking:

```
add_header X-Frame-Options "SAMEORIGIN" always;
```

As a modern alternative, CSP `frame-ancestors 'self'` provides the same protection.

3. Add X-Content-Type-Options

Prevents the browser from MIME-type sniffing and misinterpreting content (and the XSS risk it opens):

```
add_header X-Content-Type-Options "nosniff" always;
```

4. Add Strict-Transport-Security (HSTS)

Enforces HTTPS and makes SSL-stripping/MITM attacks harder. (Apply only if your site is fully HTTPS.)

```
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
```

5. Add Referrer-Policy

Reduces information leakage in the `Referer` header sent to external links:

```
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
```

6. Add Permissions-Policy

Restricts browser APIs (camera, microphone, location, etc.); prevents unwanted access by third-party iframes:

```
add_header Permissions-Policy "geolocation=(), microphone=(), camera=()" always;
```

7. X-XSS-Protection (defence in depth)

A historical header for old browsers; the real protection is in the CSP. You may add it optionally:

```
add_header X-XSS-Protection "1; mode=block" always;
```

Combined configuration — Nginx

Add it to your server block (server { ... }), validate with `nginx -t`, then reload with `systemctl reload nginx`:

```
add_header Content-Security-Policy "default-src 'self'; frame-ancestors 'self'" always;
add_header X-Frame-Options "SAMEORIGIN" always;
add_header X-Content-Type-Options "nosniff" always;
add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;
add_header Referrer-Policy "strict-origin-when-cross-origin" always;
add_header Permissions-Policy "geolocation=(), microphone=(), camera=()" always;
```

Ready-made configuration for other platforms

If your server is not Nginx, you can add the same headers with the appropriate ready-made block below.

Firebase Hosting — `firebase.json`:

```
{
  "hosting": {
    "headers": [
      {
        "source": "**",
        "headers": [
          { "key": "Content-Security-Policy", "value": "default-src 'self'; frame-ancestors 'self'" },
          { "key": "X-Frame-Options", "value": "SAMEORIGIN" },
          { "key": "X-Content-Type-Options", "value": "nosniff" },
          { "key": "Strict-Transport-Security", "value": "max-age=31536000; includeSubDomains" },
          { "key": "Referrer-Policy", "value": "strict-origin-when-cross-origin" },
          { "key": "Permissions-Policy", "value": "geolocation=(), microphone=(), camera=()" }
        ]
```

```
      }
    ]
  }
}
```

Vercel — `vercel.json`:

```
{
  "headers": [
    {
      "source": "/(.*)",
      "headers": [
          { "key": "Content-Security-Policy", "value": "default-src 'self'; frame-ancestors 'self'" },
          { "key": "X-Frame-Options", "value": "SAMEORIGIN" },
          { "key": "X-Content-Type-Options", "value": "nosniff" },
          { "key": "Strict-Transport-Security", "value": "max-age=31536000; includeSubDomains" },
          { "key": "Referrer-Policy", "value": "strict-origin-when-cross-origin" },
          { "key": "Permissions-Policy", "value": "geolocation=(), microphone=(), camera=()" }
      ]
    }
  ]
}
```

Next.js — `next.config.js`:

```
module.exports = {
  async headers() {
    return [
      {
        source: '/(.*)',
        headers: [
          { key: 'Content-Security-Policy', value: 'default-src 'self'; frame-ancestors 'self'' },
          { key: 'X-Frame-Options', value: 'SAMEORIGIN' },
          { key: 'X-Content-Type-Options', value: 'nosniff' },
          { key: 'Strict-Transport-Security', value: 'max-age=31536000; includeSubDomains' },
          { key: 'Referrer-Policy', value: 'strict-origin-when-cross-origin' },
          { key: 'Permissions-Policy', value: 'geolocation=(), microphone=(), camera=()' }
        ],
      },
    ];
  },
};
```

Apache — `.htaccess` (mod_headers):

```
Header always set Content-Security-Policy "default-src 'self'; frame-ancestors 'self'"
Header always set X-Frame-Options "SAMEORIGIN"
Header always set X-Content-Type-Options "nosniff"
Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"
Header always set Referrer-Policy "strict-origin-when-cross-origin"
Header always set Permissions-Policy "geolocation=(), microphone=(), camera=()"
```

IIS / ASP.NET — `web.config`:

```
<configuration>
  <system.webServer>
    <httpProtocol>
      <customHeaders>
      <add name="Content-Security-Policy" value="default-src 'self'; frame-ancestors 'self'" />
      <add name="X-Frame-Options" value="SAMEORIGIN" />
      <add name="X-Content-Type-Options" value="nosniff" />
      <add name="Strict-Transport-Security" value="max-age=31536000; includeSubDomains" />
      <add name="Referrer-Policy" value="strict-origin-when-cross-origin" />
      <add name="Permissions-Policy" value="geolocation=(), microphone=(), camera=()" />
      </customHeaders>
    </httpProtocol>
  </system.webServer>
</configuration>
```

After the changes, verify the headers are added to the response using the browser developer tools (Network tab) or `curl -I https://testasp.vulnweb.com`.

5. Appendix — Glossary

SQL Injection	An injection flaw where user input reaches a database query.
XSS	Cross-Site Scripting — injection of malicious scripts into pages.
IDOR	Insecure Direct Object Reference — accessing others' records via ID manipulation.
TLS	Transport Layer Security.
HSTS	HTTP Strict Transport Security header.
CSP	Content Security Policy.
CORS	Cross-Origin Resource Sharing.
Clickjacking	Tricking clicks via invisible framing.