Beagle Security

Web server security checklist

A practical companion to our web server security guide. Use this checklist to review the controls that keep your server software, network exposure, configurations, and application data protected.

Work through each category against your own environment. Tick an item only when the control is configured and verified.

System & OS security

- [ ] **Keep your operating system & server software updated**

  Apply updates to web servers, frameworks, and dependencies regularly. The blog recommends validating critical patches in staging before rolling them into production.

- [ ] **Disable unused services & ports**

  Remove services and close ports the server does not need. This reduces the number of entry points an attacker can reach.

- [ ] **Use strong passwords or SSH keys**

  Use strong authentication for administrative access. The blog also recommends multi-factor authentication and limiting administrative privileges.

- [ ] **Disable root login via SSH**

  Prevent direct root access over SSH so administrative activity is tied to controlled accounts rather than a shared, unrestricted account.

- [ ] **Restrict access to config & log files**

  Limit access to configuration and log files so sensitive settings, credentials, and operational information are not exposed to unauthorized users.

- [ ] **Automate patch management where possible**

  Automate updates where practical, while validating critical patches in staging first to reduce the risk of disrupting the running environment.

Network & access control

- [ ] **Allow only necessary traffic through firewalls**

  Use firewalls to permit only required traffic. This supports the layered network security approach described in the blog.

- [ ] **Use a web application firewall (WAF)**

  Use a WAF as an additional protection layer for web-facing traffic and application attacks.

- [ ] **Limit admin panel access to trusted IPs**

  Restrict administrative interfaces to trusted sources and combine this with strong authentication and least-privilege access.

- [ ] **Enable intrusion detection or prevention tools**

  Use IDS or IPS controls to identify unusual traffic and intrusion attempts early, then alert or block suspicious activity as appropriate.

HTTPS & TLS configuration

- [ ] **Enforce HTTPS on all pages**

  Redirect HTTP traffic to HTTPS so communication between users and the server is encrypted.

- [ ] **Disable old SSL/TLS versions**

  Disable outdated protocols such as SSLv3 and avoid weak TLS configurations.

- [ ] **Use strong ciphers & keys**

  Use secure TLS settings to protect data in transit and reduce exposure to weak encryption.

- [ ] **Enable HSTS headers**

  Use HSTS to help ensure browsers continue to use HTTPS when connecting to the server.

- [ ] **Keep SSL certificates valid & updated**

  Monitor certificate expiry and keep certificates current so encrypted connections remain trusted and available.

Server configuration

- [ ] **Remove default admin accounts**

  Remove or secure default administrative accounts so attackers cannot rely on known credentials or unnecessary privileged access.

- [ ] **Disable directory listing**

  Disable directory listing to avoid exposing files, paths, or application structure that can help an attacker map the server.

- [ ] **Set correct file permissions**

  Apply file permissions that prevent unauthorized changes to server content, configuration, and sensitive data.

- [ ] **Hide server version & banner info**

  Limit unnecessary server information in banners so attackers receive less detail about the software and versions they may target.

- [ ] **Remove unused modules or plugins**

  Remove modules and plugins that are not required. Unused components add an attack surface and can introduce known vulnerabilities.

Application & database security

- [ ] **Keep CMS, frameworks, & plugins updated**

  Keep application components current to patch known vulnerabilities in frameworks, CMS platforms, and dependencies.

- [ ] **Use secure, rotated database credentials**

  Protect database access with secure credentials and rotate them through a controlled process.

- [ ] **Limit database access to specific IPs**

  Restrict database access to approved sources so the database is not reachable more broadly than necessary.

- [ ] **Validate & sanitize user inputs**

  Validate and sanitize inputs to reduce the risk of attacks such as SQL injection, where malicious input can manipulate backend queries.

Monitoring & maintenance

- [ ] **Enable & review server logs regularly**

  Centralize and review logs to detect suspicious IPs, repeated failed logins, and unusual activity. The blog cites tools such as ELK Stack and Splunk for log management.

- [ ] **Run vulnerability scans or pen tests**

  Use automated vulnerability assessments and supplement them with manual penetration testing for deeper insight. Prioritize high-severity findings using CVSS scores.

- [ ] **Back up data & configs securely**

  Automate backups of critical server data, store them offsite or in secure cloud storage, and test recovery procedures periodically.

- [ ] **Review security settings periodically**

  Review settings after major updates, configuration changes, or new deployments, and maintain regular verification cycles.

Where Beagle Security fits

Most of what's above gets done once and forgotten until something breaks or an audit asks for proof. Vulnerability scans and penetration tests don't act that way. They need to happen repeatedly and tied to every release.

Beagle Security is built to keep that going. It runs continuous, AI driven vulnerability assessments against your web applications and servers, plugging directly into CI/CD pipelines so testing happens alongside your deployments instead of as a separate task someone has to remember to run.

Try it for free here and see how it fits into your release process.