A Style Guide for Writing About WordPress and Security — DRAFT

Principles, Practices, and Terminology for Clear, Honest, and Empowering Security Communication

Dan Knauss | February 16, 2026

1. Security, Vulnerability, and Trust in Open Source

We maintain a security product, but the product is not security.

As part of the Open Web, WordPress is a commons, and so is WordPress security.

Security is never absolute, which is to say, security always comes with vulnerability. We are always vulnerable in some way, however small. We are never completely invulnerable.

Open source leads by refusing to pretend software can ever be perfect — especially by hiding the source code. Open source means working in the open — together.

That's never easy. We must constantly resist our inclination to hide defects and vulnerabilities — to create a mask of invulnerability based on obscurity and deception.

We are confident in the security of our systems when we believe our trust is well-founded in our tools, partners, experts, and other authorities we rely on for advice and insight.

Our confidence and trust require maintenance, learning, and growth in cooperative relationships. Together, we take care of our shared tools, knowledge, and relationships — with colleagues, partners, customers, and even competitors.

Because our security and vulnerability are shared collectively, so is responsibility. If responsibility is shared, so is the quality, security, and trust it generates in our customers and marketplace.

Effective security communication must go beyond awareness to foster behavioral change. We prioritize resilience — the ability to recover quickly from inevitable breaches — as much as prevention. Our goal is to make security practices habitual and deeply embedded in our organizational culture.

2. In/vulnerability: Dilemma and Opportunity

Writing about security, especially in open source, is a tricky rhetorical situation. There are several dilemmas presented to anyone with "bad news" facing an audience of superiors, peers, customers, and competitors, especially in contexts where "professionalism" is often misconstrued as a performance or mask of invulnerability, if not omniscience.

Admitting errors, defects, new risks, and security failures may cause individuals, organizations, and brands to lose trust. But denying, hiding, or lying about security failures always fails harder in the end. It's devastating to brands, products, reputations, and careers. We see this happen time and again.

Maximizing security — and trust — in open source requires exposing all our work (warts and all) to everyone for review (or exploitation) by anyone.

3. Writing about Security and Vulnerabilities in WordPress

3.1 Lead with Solutions, Not Fear

Be as accurate as possible about threats, but orient your writing toward solutions. Security writing should inform and empower, not alarm or paralyze. The goal is to help people understand real risks and take effective, proportionate action.

> ***Guiding Principle:*** *Don't use fear, uncertainty, and doubt (FUD) to sell solutions. Dispel fear with knowledge. Demonstrate how reasonable levels of risk can be managed. Foster confidence in the tools, information, and relationships that empower WordPress users to protect their web properties.*

When describing threats, always provide an actionable path forward. What knowledge, tools, or relationships will most effectively reduce the risk? Lead with that. When writing about incidents or breaches, extract the lesson and present it constructively. There is almost always an upside in what can be learned and improved.

3.2 Be Realistic and Proportionate

Avoid both minimizing and overstating threats. Don't tell open-ended stories of vague, dark unknowns. Be specific about what the risk is, who it affects, and what can be done about it. If a vulnerability is critical, say so clearly. If a risk is low for most users, say that too.

Remember that the vast majority of WordPress compromises trace back to a few recurring causes: weak or reused passwords, outdated plugins with known vulnerabilities, and neglected server environments. This is not new information for security professionals, but it is new for many WordPress users. Repeat these messages clearly and without condescension.

3.3 Place Responsibility Where It's Constructive

When assigning responsibility, make sure it serves a purpose. Responsibility should point toward action, not blame. Be clear about what is within the user's control, what depends on their hosting provider, and what falls to the WordPress core team or plugin developers.

3.4 Make Security Accessible and Engaging

Write for the WordPress user who is still learning. Security topics can be dry, intimidating, or both. Work to make them interesting, practical, and empowering. Use clear examples, relatable scenarios, and plain language. If a concept requires technical depth, build up to it. Always define terms on first use.

3.5 Writing about AI, LLMs, and Automated Security Tools

AI-powered tools are increasingly used in WordPress security — for malware scanning, anomaly detection, code review, and vulnerability assessment. When writing about these tools, follow these principles:

- **Be specific about capabilities.** Describe what the tool does in concrete terms rather than vague claims. Avoid describing AI tools as "intelligent," "thinking," or "understanding" — they perform computation, not cognition.
- **Disclose AI-generated content.** When content, code samples, or analysis are produced with AI assistance, say so. Transparency supports trust.
- **Address AI-specific threat vectors.** AI introduces its own risks to the WordPress ecosystem. When relevant, discuss prompt injection, training-data poisoning, AI-generated phishing, deepfake-based social engineering, shadow AI, and insecure AI-generated code.
- **Avoid hype and anthropomorphism.** AI is a tool, not a colleague. Don't attribute agency, judgment, or intent to automated systems.

3.6 Writing about Compliance and Regulatory Frameworks

WordPress is often deployed in environments subject to regulatory requirements (SOC 2, PCI-DSS, HIPAA, GDPR, FedRAMP). When writing about compliance:

- **Don't claim WordPress is "compliant."** Software is not compliant; deployments are.
- **Reference the specific framework and control.** Vague claims like "meets industry standards" are unhelpful.
- **Acknowledge shared responsibility.** Compliance depends on the software, the hosting environment, and the site operator.
- **Distinguish between certification and alignment.**

4. Audience, Voice, and Tone

4.1 Know Your Audiences

Security writing in the WordPress ecosystem addresses several overlapping audiences: WordPress site owners and administrators, developers and engineers, enterprise decision-makers (CTOs, CISOs, IT directors), non-technical stakeholders (marketers, content editors, business owners), and WordPress community members and contributors.

4.2 Voice

Your voice should convey the brand's personality and values. It is:

- **Confident** — grounded in knowledge and experience, never bluffing or overpromising.
- **Candid** — honest about problems, limitations, and uncertainty.
- **Expert** — technically accurate, well-sourced, and current.
- **Accessible** — warm, clear, and human.
- **Open** — reflecting the open-source values of transparency, collaboration, and shared responsibility.

4.3 Tone

Tone adapts to context while the voice remains consistent. The default tone for security writing is realistic about problems, optimistic and reassuring, down-to-earth, and clear, concise, and honest.

> ***Tone Shift by Context:*** *Vulnerability disclosure: measured, precise, actionable. Educational content: encouraging, patient, building understanding step by step. Incident response guidance: calm, clear, directive. Thought leadership: reflective, well-sourced, open to nuance and debate.*

5. Inclusive Communication

5.1 Bring Outsiders In

Security writing can easily become a closed conversation among experts. Actively work against this. Explain jargon and technical terms the first time you use them. Spell out acronyms on first use.

5.2 Language Choices

Use inclusive, contemporary language:

Preferred	Avoid
allowlist / denylist	whitelist / blacklist
primary / replica	master / slave
brute-force attack / credential stuffing attack	brute-force hacking
threat actor	hacker (when meaning attacker)

5.3 WordPress-Specific Terminology

- **Dashboard** — the WordPress admin interface (avoid "backend" in user-facing writing).
- **Plugin** — an extension that adds functionality. Always one word, lowercase in running text.
- **Theme** — a collection of templates and stylesheets controlling visual presentation.
- **Multisite** — a WordPress feature enabling multiple sites on one installation. One word, capitalized.

- **Auto-update** — WordPress's built-in mechanism for applying updates automatically. Hyphenated.

6. Technical Formatting Guidelines

6.1 Two-Font System

Use two font treatments: normal font for names of products, organizations, document titles, and human-facing concepts; monospace font for code, commands, file paths, configuration values, and machine-facing identifiers.

6.2 When to Use Monospace

File names and paths, configuration constants and PHP functions, command-line tools and commands, database fields and table names, HTTP headers and status codes, CVE identifiers and version numbers in technical context.

> ***Boundary rule:*** *Use monospace when the version number is the point of the sentence — the reader needs to act on it. Use normal font when the version appears as background context in running prose.*

6.3 When to Use Normal Font

Product names and their versions in running prose, names of organizations and teams, concepts and roles, error messages and security prompts when quoting for a non-technical audience.

6.4 Bold and Emphasis

Use bold sparingly for key terms being defined, important warnings, and UI elements the reader needs to find and click. Use italics for emphasis, titles of publications, and introducing new terms.

6.5 Acronyms

Spell out acronyms on first use, followed by the abbreviation in parentheses. Use the acronym alone in subsequent references.

7. Writing about Vulnerabilities

7.1 Disclosure and Reporting

When writing about specific vulnerabilities, follow established responsible disclosure conventions: use the official CVE identifier, name the affected software and versions, describe the vulnerability type using standard terminology, state the severity using an established framework, provide the remediation, and credit the researcher or security team.

7.2 Severity Language

Match the urgency of your language to the actual severity of the vulnerability, using the Common Vulnerability Scoring System (CVSS):

CVSS Range	Internal Label	Default Communication Channel
9.0 - 10.0	Critical	Dedicated customer email
7.0 - 8.9	High	Dedicated customer email
4.0 - 6.9	Medium	Case-by-case decision
0.1 - 3.9	Low	Product update email

7.3 Avoiding Sensationalism

Security news attracts clicks, and sensationalism is common in the WordPress security ecosystem. Resist it. Write with precision. When writing about a vulnerability, include context signals: active install count, authentication requirement, default vs. non-default configuration, affected version range, auto-update availability, and EPSS score when available.

7.4 Core vs. Plugin vs. Theme Vulnerabilities

Adjust your writing based on whether the vulnerability is in WordPress core, a plugin, or a theme. Each has different disclosure norms, timelines, and audience expectations.

7.5 Naming Plugins and Themes

Always name the affected plugin or theme. Users cannot act on vague warnings. Exercise proportionality based on the plugin's install base. Never editorialize about a plugin author's competence or responsiveness.

7.6 Operational Policy Boundary

This style guide defines writing standards. Operational procedures are maintained separately in Section 9.

7.7 Writing about Supply Chain Incidents

When writing about supply chain attacks targeting the WordPress ecosystem, name the mechanism, distinguish intent from negligence, provide a timeline, describe the blast radius, and recommend specific actions.

8. Glossary of WordPress Security Terms

2FA / MFA — Two-factor authentication / multi-factor authentication. A security mechanism requiring two or more verification methods to access an account.

Action-gated reauthentication — A security mechanism requiring identity re-verification before performing sensitive actions. Also known as "sudo mode."

Admin (role) — The highest default user role in a single-site WordPress installation.

Application password — A feature (since WordPress 5.6) for REST API and XML-RPC authentication. Provides scoped, revocable credentials that bypass 2FA and do not expire by default.

Argon2id — A modern password hashing algorithm. bcrypt is the WordPress default (since 6.8); Argon2id can be enabled via the wp_hash_password filter.

Attack surface — The total set of points where an attacker can attempt to enter or extract data from a system.

Auth cookie — The signed session cookie WordPress sets when a user logs in.

Auto-update — WordPress's built-in mechanism for automatically applying updates. Minor (security) releases auto-update by default since 3.7.

bcrypt — The default password hashing algorithm in WordPress since version 6.8, with SHA-384 pre-hashing. BLAKE2b via Sodium is used for application passwords and tokens.

Brute-force attack — An attack method that systematically tries many credential combinations.

Capability — A specific permission assigned to a WordPress user role.

Content Security Policy (CSP) — An HTTP response header controlling which resources a browser may load.

Credential stuffing — An automated attack using leaked username/password pairs from other breaches.

Cross-Site Request Forgery (CSRF) — An attack tricking authenticated users into performing unintended actions. WordPress mitigates CSRF through nonces.

Cross-Site Scripting (XSS) — A vulnerability allowing injection of malicious scripts into web pages viewed by other users.

CVE — Common Vulnerabilities and Exposures. A standardized identifier for publicly disclosed vulnerabilities.

CVSS — Common Vulnerability Scoring System. A framework for rating vulnerability severity on a 0-10 scale.

Dashboard — The WordPress administrative interface. Prefer "Dashboard" over "backend" or "admin panel."

Deepfake — Synthetic media generated by AI to impersonate a real person. Found in 35% of AI-driven breaches (IBM 2025).

Dependency confusion — A supply chain attack exploiting public/private package name conflicts.

EPSS — Exploit Prediction Scoring System. Estimates the probability a vulnerability will be exploited within 30 days.

Fail2Ban — A server-level intrusion prevention tool that monitors logs and bans malicious IPs.

FedRAMP — Federal Risk and Authorization Management Program for cloud products and services.

File integrity monitoring — Detecting unauthorized file changes by comparing checksums against a known-good baseline.

FUD — Fear, uncertainty, and doubt. A rhetorical strategy to avoid in security writing.

Hardening — Reducing a system's attack surface by disabling unnecessary features and restricting permissions.

HSTS — HTTP Strict Transport Security. Instructs browsers to connect only over HTTPS.

Infostealer — Malware designed to exfiltrate passwords, session cookies, and browser data. 30% of infected systems were enterprise-licensed (Verizon DBIR 2025).

IoC (Indicators of Compromise) — Observable evidence of system compromise.

Malware — Malicious software. In WordPress: injected PHP backdoors, JavaScript redirects, SEO spam, cryptominers, and phishing pages.

Multisite — A WordPress feature for running multiple sites from a single installation.

Nonce — A WordPress cryptographic token for CSRF protection. Despite the name, WordPress nonces are valid for up to 24 hours, not single-use.

OWASP Top 10 — The ten most critical web application security risks. Current edition: 2025.

Passkey / WebAuthn — A passwordless authentication standard based on public-key cryptography. Resists phishing, credential stuffing, and replay attacks.

Patch / Patching — A software update fixing a bug or vulnerability. "Virtual patching" refers to WAF rules blocking exploitation before a code-level fix.

Phishing — Social engineering using deceptive communications. "Spear phishing" targets individuals; "whaling" targets executives.

Plugin — A software extension adding functionality to WordPress. Always one word, lowercase.

PoC (Proof of Concept) — A demonstration proving a vulnerability is exploitable.

Principle of Least Privilege (PoLP) — Granting only the minimum permissions necessary.

Rate limiting — Restricting the number of requests a client can make within a time window.

Responsible disclosure — Reporting a vulnerability privately before public disclosure.

REST API — WordPress's built-in API for programmatic access to site data.

Role — A named collection of capabilities in WordPress.

SBOM (Software Bill of Materials) — A machine-readable record of all software components and dependencies.

Shadow AI — Unsanctioned employee use of AI tools. Added $200K to average breach costs (IBM 2025).

Session hijacking — Obtaining a valid session cookie to impersonate an authenticated user. 2FA does not protect against hijacked sessions.

SQL injection (SQLi) — Inserting malicious SQL into database queries. WordPress mitigates via $wpdb->prepare().

SSRF (Server-Side Request Forgery) — Causing the server to make HTTP requests to unintended destinations. Classified under A01 (Broken Access Control) in OWASP Top 10:2025.

Supply chain attack — Compromising software through its dependencies or distribution channels.

Theme — Template files and stylesheets controlling a WordPress site's visual presentation.

Threat actor — An individual or group exploiting vulnerabilities for malicious purposes.

TOTP — Time-based One-Time Password (RFC 6238). Codes typically valid for 30 seconds.

Virtual patching — WAF rules blocking exploitation of known vulnerabilities at the network level.

Vulnerability — A weakness that could be exploited to compromise security.

WAF — Web Application Firewall. Filters and monitors HTTP traffic.

XML-RPC — A legacy remote procedure call protocol. Common target for brute-force amplification.

Zero-day — A vulnerability that is publicly unknown and unpatched.

Zero Trust — A security model requiring continuous verification of all users and devices regardless of network location.

9. Operational Appendix: Vulnerability Communication Workflow

Development provides Customer Success and Marketing with vulnerability description and classification, CVE details, patch details, timeline and scope, discovery and attribution, exploitation history, and additional context.

External Workflow:

1. Preparation: Development provides the information listed above.
2. Drafting: Customer Success drafts communications for leadership approval.
3. Coordination: Development releases the patch and notifies Customer Success and Marketing.
4. PSA Release Timing: Determine timeframe based on severity and disclosure status.
5. Execution: Marketing communicates via established templates and channels.

10. References and Further Reading

Style and Writing Guides: Bishop Fox Cybersecurity Style Guide, Google Developer Documentation Style Guide, Microsoft Writing Style Guide.

Security Terminology References: NIST Glossary, OWASP Glossary, SANS Security Glossary.

WordPress Security Resources: WordPress Security White Paper, Hardening WordPress, Patchstack, Wordfence Intelligence, WPScan.

Related Documents

- **WordPress Security Architecture and Hardening Guide** — Enterprise-focused security architecture.
- **WordPress Security Benchmark** — Prescriptive, auditable hardening controls.
- **WordPress Security White Paper** (WordPress.org, September 2025) — Official upstream document.

Terminology and formatting conventions adapted from the Bishop Fox Cybersecurity Style Guide (2023). AI threat data from the Verizon DBIR (2025) and IBM Cost of a Data Breach Report (2025).