WHITE PAPER

Common Security Fixes

A Remediation Handbook for Development and Security Teams

Concrete, implementation-ready remediation guidance for the most frequently identified vulnerabilities across web applications, APIs, network infrastructure, and cloud environments.

Version 1.0

Prepared By Lorikeet Security

lorikeetsecurity.com

OVERVIEW

1. Introduction

Identifying vulnerabilities is only half the work. Effective remediation requires clear, actionable guidance that engineering and operations teams can act on immediately. This handbook provides implementation-ready fixes for the vulnerabilities most commonly identified by Lorikeet Security across hundreds of assessments.

Each fix is organized by severity and includes the affected component, root cause, and the specific remediation action required. All guidance aligns to OWASP, NIST SP 800-53, and applicable compliance framework requirements.

CRITICAL

2. Critical Severity Fixes

Vulnerability	Root Cause	Fix
SQL Injection	User input concatenated into SQL queries without parameterization.	Use parameterized queries or prepared statements in all database interactions. Never construct queries by string concatenation. Example: use PDO with bindParam() in PHP, or SQLAlchemy's parameterized queries in Python.
Authentication Bypass	Server-side auth checks absent or bypassable via parameter manipulation, HTTP method switching, or forced browsing.	Implement server-side session validation on every request. Never trust client-supplied role or permission claims. Use framework-level auth middleware applied globally.
IDOR (Broken Object Auth)	API returns or modifies resources based on user-supplied ID without verifying ownership.	On every resource access, verify that the authenticated user owns or has explicit permission to access the requested object. Enforce this server-side, not just in the UI.
JWT None Algorithm / Weak Secret	JWT library accepts "none" algorithm or verifies with a weak, guessable secret.	Explicitly whitelist accepted algorithms (e.g., RS256). Reject tokens with "none" algorithm. Use secrets ≥256 bits, generated by a CSPRNG and stored in a secrets manager.
Hardcoded Secrets in Code / Repos	API keys, passwords, or tokens committed to source code or present in frontend bundles.	Use environment variables or a secrets manager (AWS Secrets Manager, HashiCorp Vault). Rotate all exposed secrets immediately. Add pre-commit hooks using tools like trufflehog or gitleaks.
SSRF via User-Controlled URLs	Application fetches user-supplied URLs without restriction, enabling access to internal metadata endpoints.	Implement a strict allowlist of permitted URL schemes and destinations. Block access to RFC1918 address space and cloud metadata IPs (169.254.169.254). Use a dedicated, sandboxed HTTP client for external fetches.

Vulnerability	Root Cause	Fix
Publicly Accessible Cloud Storage	S3 buckets or Azure Blob containers publicly accessible due to misconfigured ACLs.	Enable Block Public Access at the AWS account level. Audit all bucket policies. Require bucket-level encryption. Use CSPM tooling for continuous misconfiguration detection.

HIGH

3. High Severity Fixes

Vulnerability	Root Cause	Fix
Missing MFA on Privileged Access	Administrative and remote access authenticated with passwords alone.	Enforce TOTP or FIDO2/WebAuthn MFA on all admin consoles, VPN access, and cloud management portals. Remove fallback SMS-only MFA for privileged accounts.
Insecure TLS Configuration	Deprecated protocols (TLS 1.0/1.1) or weak cipher suites enabled.	Disable TLS 1.0 and 1.1. Enforce TLS 1.2 minimum (TLS 1.3 preferred). Use cipher suite: TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256. Enable HSTS (min-age: 31536000).
Stored XSS	User-supplied HTML or JavaScript stored and rendered without sanitization.	Implement context-aware output encoding for all user-controlled values. Deploy a strict Content Security Policy. Use DOMPurify for any HTML rendering. Avoid innerHTML in favor of textContent.
Mass Assignment	API accepts undocumented fields in requests, allowing users to set role, admin flags, or pricing fields.	Define explicit allow-lists of permitted fields for each API endpoint. Never bind request payloads directly to data models. Use dedicated request DTOs with validated field sets.
Overly Permissive CORS	Access-Control-Allow-Origin set to wildcard (*) or reflecting arbitrary origins with credentials.	Implement a strict origin allowlist. Never combine Access-Control-Allow-Credentials: true with a wildcard origin. Validate the Origin header server-side before setting CORS headers.
Subdomain Takeover	DNS CNAME records pointing to deprovisioned cloud resources (e.g., expired Azure/GitHub/Heroku endpoints).	Audit all DNS CNAME records. Remove or redirect dangling records. Implement continuous DNS monitoring using tools like Subfinder or nuclei-based detection.

Vulnerability	Root Cause	Fix
Unpatched Libraries (CVEs)	Third-party dependencies with known critical or high CVEs in production.	Implement Software Composition Analysis (SCA) in CI/CD pipelines using tools like Snyk or OWASP Dependency-Check. Define SLAs for patching critical CVEs (≤15 days). Pin dependency versions and track changelogs.

MEDIUM

4. Medium Severity Fixes

Vulnerability	Fix
Missing CSRF Protection	Implement synchronizer token pattern or use SameSite=Strict/Lax cookies. Verify the Origin and Referer headers on state-changing requests.
Missing Security Headers	Add: Strict-Transport-Security, X-Content-Type-Options: nosniff, X-Frame-Options: DENY, Referrer-Policy: strict-origin-when-cross-origin, Permissions-Policy.
Excessive API Response Verbosity	Audit all API response schemas. Remove internal fields, system metadata, and PII not required by the consuming application. Use response DTOs rather than serializing full model objects.
Missing Rate Limiting on Auth Endpoints	Implement rate limiting on login, registration, password reset, and OTP endpoints. Use exponential backoff + account lockout after N failed attempts. Use a dedicated rate-limiting middleware (e.g., express-rate-limit, rack-attack).
Session Fixation	Regenerate the session ID immediately upon authentication. Ensure the pre-login session ID is invalidated on login. Use a server-side session store.
Clickjacking Exposure	Add X-Frame-Options: DENY or use CSP frame-ancestors 'none'. Apply to all application pages, not just the login page.
HTTP Verb Tampering	Disable HTTP method override headers on the server. Restrict API endpoints to their intended HTTP methods. Return 405 Method Not Allowed for unexpected verbs.
Insecure File Upload	Validate MIME type server-side using file content inspection (not extension or Content-Type header). Store uploads outside the web root. Generate random filenames. Scan for malware before serving.

CLOUD & INFRA

5. Cloud and Infrastructure Fixes

Area	Fix
AWS IAM Overprivilege	Apply least privilege to all IAM users, roles, and policies. Remove wildcard resource permissions. Use IAM Access Analyzer to detect unused permissions. Enforce MFA for console access.
Exposed EC2 Management Ports	Remove RDP (3389) and SSH (22) exposure from security groups. Route admin access through AWS Systems Manager Session Manager or a VPN with MFA.
Unencrypted S3 Data	Enable S3 SSE-S3 or SSE-KMS encryption at rest for all buckets. Use S3 Bucket Policies to deny unencrypted uploads (condition: s3:x-amz-server-side-encryption).
Public RDS/Database Instances	Set PubliclyAccessible=false on all RDS instances. Place databases in private subnets. Restrict security group access to application subnet CIDR only.
CloudTrail / Logging Gaps	Enable CloudTrail in all regions with S3 log integrity validation. Enable VPC Flow Logs. Set log retention to minimum 365 days. Alert on critical API calls using CloudWatch alarms.
Container / K8s Misconfigurations	Disable privileged container execution. Enforce PodSecurityAdmission policies. Restrict hostPath mounts. Use network policies to limit pod-to-pod communication. Scan images with Trivy or Grype.

PROCESS

6. Building Remediation into Your SDLC

- Shift left: Add SAST scanning (Semgrep, CodeQL) and SCA scanning (Snyk) to your CI/CD pipeline. Catch vulnerabilities before they reach production.
- Define SLAs: Critical ≤15 days, High ≤30 days, Medium ≤90 days. Track remediation in your vulnerability management system.
- Validate fixes: Never close a finding without verification. Request retesting for Critical and High findings — Lorikeet includes this free within 30 days.
- Track regression: Re-scan after each major release to detect reintroduced vulnerabilities.
- Build a security champion program: Embed security-aware developers in each engineering team to review security findings and own remediation.

> **Remediation Support Available**
>
> Lorikeet Security's findings remediation service provides hands-on guidance to engineering teams working through complex vulnerabilities. Contact us at lorikeetsecurity.com.