

Vulnerability & Patch Management Program

RichesReach Security Operations

Last Updated: 2026

Owner: Security Team

Review Frequency: Quarterly

Overview

RichesReach maintains a proactive vulnerability management program to identify, assess, and remediate security vulnerabilities in our infrastructure, applications, and dependencies.

Vulnerability Sources

1. Dependency Scanning

- **Tool:** Dependabot (GitHub) + Snyk
- **Frequency:** Daily automated scans
- **Scope:** Python packages, Node.js packages, Docker images
- **Integration:** Automated PR creation for updates

2. Container Scanning

- **Tool:** AWS ECR image scanning + Trivy
- **Frequency:** On every image build

- **Scope:** Base images, installed packages, OS-level vulnerabilities
- **Integration:** CI/CD pipeline blocking on critical vulnerabilities

3. Infrastructure Scanning

- **Tool:** AWS Security Hub + Inspector
- **Frequency:** Weekly automated scans
- **Scope:** EC2 instances, RDS, ECS containers
- **Integration:** CloudWatch alarms for critical findings

4. External Advisories

- **Sources:**
 - CVE/NVD (National Vulnerability Database)
 - GitHub Security Advisories
 - AWS Security Bulletins
 - Django Security Advisories
 - Python Security Advisories
- **Monitoring:** Automated RSS feeds + manual review

Severity Classification

Severity	CVSS Score	Description	Example
Critical	9.0-10.0	Remote code execution, data breach, authentication bypass	Log4Shell, Heartbleed
High	7.0-8.9	Privilege escalation, sensitive data exposure	SQL injection, XSS
Medium	4.0-6.9	Information disclosure, denial of service	Path traversal, CSRF

Severity	CVSS Score	Description	Example
Low	0.1-3.9	Limited impact, requires specific conditions	Information leakage

Patch SLAs

Severity	Target SLA	Maximum SLA	Escalation
Critical	24 hours	48 hours	CTO + Security Lead
High	7 days	14 days	Security Lead
Medium	30 days	60 days	Engineering Lead
Low	90 days	180 days	Next release cycle

Remediation Process

1. Detection

- Automated tools create tickets/PRs
- Security team reviews and triages
- Assign severity and SLA

2. Assessment

- Determine exploitability in our environment
- Assess business impact
- Check for workarounds

3. Remediation

- Apply patch or update dependency
- Test in staging environment
- Deploy to production within SLA

4. Verification

- Re-scan to confirm fix
- Update vulnerability tracking
- Document in security log

Exception Process

When to Request Exception: - Patch breaks critical functionality - No patch available (workaround required) - Business justification for delay

Process: 1. Document risk acceptance 2. Implement compensating controls 3. Set review date (max 90 days) 4. Get approval from Security Lead

Tools & Automation

Dependabot Configuration

```
# .github/dependabot.yml
version: 2
updates:
  - package-ecosystem: "pip"
    directory: "/deployment_package/backend"
    schedule:
      interval: "daily"
    open-pull-requests-limit: 10
    labels:
```

- "security"
- "dependencies"

CI/CD Integration






```
# .github/workflows/security-scan.yml
- name: Run Snyk scan
  uses: snyk/actions/python@master
  env:
    SNYK_TOKEN: ${ secrets.SNYK_TOKEN }
  with:
    args: --severity-threshold=high
```

Metrics & Reporting

Monthly Metrics: - Total vulnerabilities detected - Vulnerabilities remediated - Average time to patch (by severity) - SLA compliance rate

Quarterly Review: - Program effectiveness assessment - Tool accuracy review - Process improvements - Training needs

Evidence

For Security Questionnaires: -  Dependabot PR history (screenshot) -  Snyk dashboard (vulnerability trends) -  AWS Security Hub findings -  Patch deployment logs -  Exception requests (if any)

Sample Evidence:

```
Dependabot Alerts (Last 30 Days):
- Critical: 2 (patched within 24h)
- High: 8 (patched within 7d)
- Medium: 15 (patched within 30d)
```

SLA Compliance: 100% (all patches within target)

Contact

Security Team: security@richesreach.com

Emergency: security-incident@richesreach.com

On-Call: See PagerDuty rotation

Next Review Date: [Quarterly]

Last Updated: 2026-01-XX