# Cybersecurity Operational Plan: Manager and Team Lead Implementation Guide

Chief Information Security Officer

2026-01-02

## Executive Summary for Managers

This document provides a **practical implementation guide** for cybersecurity measures at the team and department level. Based on the security audit conducted on December 30, 2025, critical vulnerabilities have been identified that require **immediate operational action** from middle management and team leads.

> ⚠ **CRITICAL ALERT**: Data exposure vulnerability discovered. Your teams must implement protective measures within 72 hours.

### Key Management Priorities

| Priority | Task | Responsible Team | Deadline |
|---|---|---|---|
| **Critical** | Block vulnerable API endpoints | Backend/DevOps teams | 24 hours |
| 🟠 **High** | Implement security headers | Frontend teams | 48 hours |
| **Medium** | Deploy monitoring systems | SRE/Ops teams | 1 week |
| ℹ **Low** | Team security training | All teams | 2 weeks |

### Team Responsibility Matrix

**Immediate Actions (0-72 hours):**

- **Backend teams**: Close REST API vulnerabilities
- **DevOps teams**: Configure secure server settings
- **QA teams**: Validate security fixes

**Short-term Tasks (1-4 weeks):**

- **Frontend teams**: Implement CSP and SRI
- **Infrastructure teams**: Set up security monitoring
- **Security teams**: Establish incident response procedures

## Technical Situation and Team Priorities

### Critical Vulnerabilities by Team

#### Backend/API Teams - CRITICAL PRIORITY

**Issue**: Personal data exposure through REST API

- **Vulnerability**: /wp-json/wp/v2/users accessible without authentication
- **Risk**: All user data publicly accessible

- **CVSS Score**: 9.1 (Critical)

**Immediate Actions:**

```
// URGENT: Add to functions.php
add_filter('rest_endpoints', function($endpoints) {
  if (isset($endpoints['/wp/v2/users'])) {
    unset($endpoints['/wp/v2/users']);
  }
  return $endpoints;
});
```

**Backend Team Lead Checklist:**

- ☐ Immediately block access to /wp-json/wp/v2/users
- ☐ Audit all REST API endpoints
- ☐ Implement authentication for sensitive endpoints
- ☐ Set up API request logging
- ☐ Test changes on staging environment

### Frontend Teams - HIGH PRIORITY

**Issue**: Missing basic security headers

- **Vulnerabilities**: No CSP, X-Frame-Options, SRI
- **Risk**: XSS attacks, clickjacking, supply chain attacks

**Practical Solutions:**

1. **Content Security Policy (CSP)**

```
<!-- Add to <head> -->
<meta http-equiv="Content-Security-Policy"
    content="default-src 'self';
          script-src 'self' 'unsafe-inline' https://cdnjs.cloudflare.com;
          style-src 'self' 'unsafe-inline' https://fonts.googleapis.com;">
```

2. **Subresource Integrity (SRI)**

```
<!-- For all external resources -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
      integrity="sha384-
vtXRMe3mGCbOeY7l30aIg8H9p3GdeSe4IFlP6G8JMa7o7lXvnz3GFKzPxzJdPfGK"
      crossorigin="anonymous"></script>
```

**Frontend Team Lead Checklist:**

- ☐ Implement CSP headers
- ☐ Add SRI for all external resources
- ☐ Configure X-Frame-Options: DENY
- ☐ Test browser compatibility
- ☐ Update CI/CD pipeline for header validation

### DevOps/Infrastructure Teams - HIGH PRIORITY

**Issue**: Insecure web server configuration

- **Vulnerabilities**: Missing HSTS, software version disclosure
- **Risk**: Man-in-the-middle attacks, information reconnaissance

**Server Configurations:**

**Apache (.htaccess):**

```
<IfModule mod_headers.c>
   # Basic security headers
   Header always set X-Content-Type-Options "nosniff"
   Header always set X-Frame-Options "DENY"
   Header always set X-XSS-Protection "1; mode=block"
   Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"

   # Hide server information
   Header always unset Server
   Header always unset X-Powered-By
</IfModule>

# Disable server signature
ServerTokens Prod
ServerSignature Off
```

**Nginx:**

```
server {
   # Security headers
   add_header X-Content-Type-Options "nosniff" always;
   add_header X-Frame-Options "DENY" always;
   add_header X-XSS-Protection "1; mode=block" always;
   add_header Strict-Transport-Security "max-age=31536000; includeSubDomains" always;

   # Hide version
   server_tokens off;
   more_clear_headers Server;
}
```

**DevOps Team Lead Checklist:**

- ☐ Apply security configurations
- ☐ Enforce HTTPS
- ☐ Hide server software versions
- ☐ Set up security logging
- ☐ Test configurations

# Tools and Technologies

## Essential Tools for Teams

### Monitoring and Alerting

**1. Security Log Monitoring**

```
# Configure rsyslog for security events
echo "local0.*    /var/log/security.log" >> /etc/rsyslog.conf
systemctl restart rsyslog
```

**2. Configuration File Monitoring**

```
#!/bin/bash
# security_monitor.sh - Monitor configuration changes
inotifywait -m /etc/apache2/ /etc/nginx/ -e modify,create,delete \
   --format '%w%f %e %T' --timefmt '%Y-%m-%d %H:%M:%S' \
   >> /var/log/config_changes.log
```

**3. Automated Header Validation**

```
#!/usr/bin/env python3
# header_check.py - Security header validation
import requests
import sys

def check_security_headers(url):
   required_headers = {
      'X-Content-Type-Options': 'nosniff',
      'X-Frame-Options': ['DENY', 'SAMEORIGIN'],
      'Strict-Transport-Security': 'max-age=',
      'Content-Security-Policy': 'default-src'
   }

   try:
      response = requests.get(url, timeout=10)
      headers = response.headers

      print(f"Checking headers for {url}:")
      for header, expected in required_headers.items():
         if header in headers:
            print(f"   {header}: {headers[header]}")
         else:
            print(f"x {header}: MISSING")

   except Exception as e:
      print(f"Error: {e}")

if __name__ == "__main__":
   if len(sys.argv) != 2:
```

```
        print("Usage: python3 header_check.py <url>")
        sys.exit(1)

    check_security_headers(sys.argv[1])
```

## Development Tools

### 1. Pre-commit Security Hooks

```yaml
# .pre-commit-config.yaml
repos:
  - repo: https://github.com/PyCQA/bandit
    rev: 1.7.4
    hooks:
      - id: bandit
        args: ['-r', '.']

  - repo: https://github.com/Yelp/detect-secrets
    rev: v1.4.0
    hooks:
      - id: detect-secrets
        args: ['--baseline', '.secrets.baseline']
```

### 2. Automated SRI Generation

```javascript
// sri-generator.js - Automatic SRI generation
const crypto = require('crypto');
const fs = require('fs');
const https = require('https');

function generateSRI(url) {
    return new Promise((resolve, reject) => {
        https.get(url, (response) => {
            let data = '';
            response.on('data', chunk => data += chunk);
            response.on('end', () => {
                const hash = crypto.createHash('sha384').update(data).digest('base64');
                resolve(`sha384-${hash}`);
            });
        }).on('error', reject);
    });
}

// Usage in build process
async function updateSRI() {
    const externalResources = [
        'https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js',
        'https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700'
    ];
```

```
    for (const url of externalResources) {
        try {
            const sri = await generateSRI(url);
            console.log(`${url}: integrity="${sri}"`);
        } catch (error) {
            console.error(`Error for ${url}:`, error.message);
        }
    }
}

updateSRI();
```

## CI/CD Integration

**GitLab CI Configuration:**

```
# .gitlab-ci.yml - Security in CI/CD
stages:
  - security-check
  - build
  - test
  - deploy

security_headers_check:
  stage: security-check
  script:
    - python3 scripts/header_check.py $CI_ENVIRONMENT_URL
  only:
    - main
    - develop

dependency_check:
  stage: security-check
  script:
    - npm audit --audit-level moderate
    - composer audit
  allow_failure: false

sri_validation:
  stage: security-check
  script:
    - node scripts/sri-generator.js
    - git diff --exit-code # Check that SRI hashes are current
  only:
    - main
```

**GitHub Actions Configuration:**

```
# .github/workflows/security.yml
name: Security Checks
```

```yaml
on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  security-scan:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Security Header Check
      run: |
        python3 scripts/header_check.py https://staging.example.com

    - name: Dependency Audit
      run: |
        npm audit --audit-level moderate

    - name: SAST Scan
      uses: github/super-linter@v4
      env:
        DEFAULT_BRANCH: main
        GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
```

# Processes and Procedures

## Incident Response Procedures

### Level 1: Detection (Development Teams)

**Incident Indicators:**

- Unusual activity in logs
- Monitoring system alerts
- User reports of suspicious behavior
- Automated security check failures

**Immediate Actions (first 15 minutes):**

1. **Stay calm** - document all actions
2. **Isolate** - disable suspicious services
3. **Notify** - inform team lead and security team
4. **Preserve** - take snapshots of logs and system state

**Notification Template:**

SECURITY INCIDENT
Time: [YYYY-MM-DD HH:MM]
Discovered by: [Name]
System: [System/service name]
Description: [Brief problem description]
Actions taken: [What has been done]
Status: [Active/Contained/Resolved]

### Level 2: Analysis (Team Leads)

**Analysis Procedure (30-60 minutes):**

1. **Information Gathering**

```
# Incident data collection script
#!/bin/bash
INCIDENT_ID=$(date +%Y%m%d_%H%M%S)
INCIDENT_DIR="/var/log/incidents/$INCIDENT_ID"

mkdir -p $INCIDENT_DIR

# Collect logs
cp /var/log/apache2/access.log $INCIDENT_DIR/
cp /var/log/apache2/error.log $INCIDENT_DIR/
cp /var/log/security.log $INCIDENT_DIR/

# System state
ps aux > $INCIDENT_DIR/processes.txt
netstat -tulpn > $INCIDENT_DIR/network.txt
df -h > $INCIDENT_DIR/disk_usage.txt

# Recent changes
find /var/www -type f -mtime -1 > $INCIDENT_DIR/recent_changes.txt

echo "Incident data collected in $INCIDENT_DIR"
```

2. **Incident Classification**

- **Level 1**: Informational (logging, monitoring)
- **Level 2**: Warning (potential threat)
- **Level 3**: Critical (active attack, data breach)

3. **Escalation Decisions**

Level 1 → Team Lead → Documentation
Level 2 → Team Lead + Security → Enhanced monitoring
Level 3 → Immediate escalation → Management + external experts

## Security Update Procedures

### Weekly Security Checks (Every Monday)

**Team Lead Checklist:**

## Weekly Security Review

### System Updates
- [ ] Check for available security updates
- [ ] Schedule critical patch installation
- [ ] Update project dependencies (npm, composer, pip)

### Monitoring
- [ ] Review security logs for the week
- [ ] Analyze monitoring alerts
- [ ] Verify backup functionality

### Configurations
- [ ] Validate security headers
- [ ] Check SSL certificate expiration
- [ ] Audit user access permissions

### Team
- [ ] Discuss security incidents in retrospective
- [ ] Plan security training
- [ ] Update procedure documentation

### Monthly Security Audits

**Department Manager Procedure:**

1. **Technical Audit**

```
#!/bin/bash
# monthly_security_audit.sh
echo "=== Monthly Security Audit ==="
echo "Date: $(date)"

# Check users with sudo privileges
echo "Users with sudo privileges:"
grep -Po '^sudo.+:\K.*$' /etc/group

# Check open ports
echo "Open ports:"
nmap -sT -O localhost

# Check failed login attempts
echo "Failed login attempts this month:"
grep "Failed password" /var/log/auth.log | wc -l

# Check security log size
echo "Security log size:"
du -sh /var/log/security.log
```

2. **Team Process Audit**

- Security procedure compliance

- Incident response effectiveness
- Documentation quality
- Team knowledge level

# Monitoring and Control

## Monitoring Systems for Teams

### Basic Monitoring (Required for All Teams)

**1. Availability and Performance Monitoring**

```
#!/bin/bash
# basic_monitoring.sh - Basic monitoring
WEBSITE="https://example.com"
LOG_FILE="/var/log/monitoring.log"

# Check availability
if curl -s --head $WEBSITE | head -n 1 | grep -q "200 OK"; then
   echo "$(date): $WEBSITE - OK" >> $LOG_FILE
else
   echo "$(date): $WEBSITE - DOWN" >> $LOG_FILE
   # Send alert
   echo "Website is down!" | mail -s "ALERT: Website Down" admin@company.com
fi

# Check security headers
HEADERS=$(curl -s -I $WEBSITE)
if echo "$HEADERS" | grep -q "X-Content-Type-Options"; then
   echo "$(date): Security headers - OK" >> $LOG_FILE
else
   echo "$(date): Security headers - MISSING" >> $LOG_FILE
fi
```

**2. Security Log Monitoring**

```
#!/bin/bash
# log_monitor.sh - Monitor suspicious activity
SECURITY_LOG="/var/log/security.log"
ALERT_EMAIL="security@company.com"

# Search for suspicious patterns
SUSPICIOUS_PATTERNS=(
   "union.*select"
   "<script"
   "\.\./"
   "eval\("
   "base64_decode"
)
```

```
for pattern in "${SUSPICIOUS_PATTERNS[@]}"; do
    if grep -i "$pattern" $SECURITY_LOG | tail -100 | grep -q "$(date +%Y-%m-%d)"; then
        echo "Suspicious activity detected: $pattern" | \
        mail -s "ALERT: Suspicious Activity" $ALERT_EMAIL
    fi
done
```

### Advanced Monitoring (For DevOps Teams)

**1. Prometheus + Grafana Configuration**

```
# prometheus.yml
global:
  scrape_interval: 15s

scrape_configs:
  - job_name: 'security-metrics'
    static_configs:
      - targets: ['localhost:9090']
    metrics_path: /metrics
    scrape_interval: 30s

  - job_name: 'web-security'
    static_configs:
      - targets: ['example.com:443']
    metrics_path: /security-check
    scheme: https
```

**2. Custom Security Metrics**

```
#!/usr/bin/env python3
# security_metrics.py - Custom metrics for Prometheus
from prometheus_client import start_http_server, Counter, Gauge
import time
import requests
import re

# Metrics
failed_logins = Counter('failed_logins_total', 'Total failed login attempts')
security_headers = Gauge('security_headers_present', 'Security headers present', ['header'])
response_time = Gauge('security_check_response_time', 'Security check response time')

def check_security_headers(url):
    """Check security headers"""
    try:
        response = requests.get(url, timeout=10)
        headers = response.headers

        # Check for important headers
        security_headers.labels(header='x-content-type-options').set(
```

```
            1 if 'X-Content-Type-Options' in headers else 0
        )
        security_headers.labels(header='x-frame-options').set(
            1 if 'X-Frame-Options' in headers else 0
        )
        security_headers.labels(header='strict-transport-security').set(
            1 if 'Strict-Transport-Security' in headers else 0
        )

        return response.elapsed.total_seconds()
    except Exception as e:
        print(f"Check error: {e}")
        return 0

def count_failed_logins():
    """Count failed login attempts"""
    try:
        with open('/var/log/auth.log', 'r') as f:
            content = f.read()
            failed_count = len(re.findall(r'Failed password', content))
            failed_logins._value._value = failed_count
    except Exception as e:
        print(f"Log reading error: {e}")

if __name__ == '__main__':
    # Start HTTP server for metrics
    start_http_server(8000)

    while True:
        # Update metrics every 60 seconds
        response_time_val = check_security_headers('https://example.com')
        response_time.set(response_time_val)

        count_failed_logins()

        time.sleep(60)
```

## Dashboards and Reporting

### Weekly Reports for Team Leads

**Automated Report Generator:**

```
#!/usr/bin/env python3
# weekly_security_report.py
import datetime
import json
from collections import defaultdict

def generate_weekly_report():
```

```python
    """Generate weekly security report"""

    # Report period
    end_date = datetime.datetime.now()
    start_date = end_date - datetime.timedelta(days=7)

    report = {
        'period': f"{start_date.strftime('%Y-%m-%d')} - {end_date.strftime('%Y-%m-%d')}",
        'summary': {},
        'incidents': [],
        'metrics': {},
        'recommendations': []
    }

    # Analyze security logs
    security_events = analyze_security_logs(start_date, end_date)
    report['summary']['security_events'] = len(security_events)

    # Check header compliance
    headers_status = check_headers_compliance()
    report['metrics']['headers_compliance'] = headers_status

    # Recommendations
    if security_events:
        report['recommendations'].append("Security events detected - analysis required")

    if not headers_status['all_present']:
        report['recommendations'].append("Not all security headers are configured")

    return report

def analyze_security_logs(start_date, end_date):
    """Analyze security logs for period"""
    events = []
    try:
        with open('/var/log/security.log', 'r') as f:
            for line in f:
                # Simple parsing - real implementation needs more complexity
                if 'SECURITY' in line:
                    events.append(line.strip())
    except FileNotFoundError:
        pass

    return events

def check_headers_compliance():
    """Check security header compliance"""
    import requests
```

```python
    try:
        response = requests.get('https://example.com', timeout=10)
        headers = response.headers

        required_headers = [
            'X-Content-Type-Options',
            'X-Frame-Options',
            'Strict-Transport-Security',
            'Content-Security-Policy'
        ]

        present = [h for h in required_headers if h in headers]

        return {
            'total_required': len(required_headers),
            'present': len(present),
            'missing': [h for h in required_headers if h not in present],
            'all_present': len(present) == len(required_headers)
        }
    except Exception as e:
        return {'error': str(e)}

if __name__ == '__main__':
    report = generate_weekly_report()

    # Save report
    filename = f"security_report_{datetime.datetime.now().strftime('%Y%m%d')}.json"
    with open(filename, 'w', encoding='utf-8') as f:
        json.dump(report, f, indent=2, ensure_ascii=False)

    print(f"Report saved: {filename}")

    # Send via email (optional)
    # send_report_email(report)
```

## Team Collaboration and Delegation

### Responsibility Matrix (RACI)

| Task | Backend | Frontend | DevOps | QA | Security |
|---|---|---|---|---|---|
| **Block API vulnerabilities** | R | I | C | A | C |
| **Implement CSP/SRI** | I | R | C | A | C |
| **Configure server headers** | I | I | R | A | C |
| **Security monitoring** | I | I | R | I | A |
| **Team training** | C | C | C | C | R |
| **Incident response** | C | C | C | C | R |

**Legend:**

- **R** (Responsible) - Responsible for execution
- **A** (Accountable) - Accountable for results
- **C** (Consulted) - Consulted
- **I** (Informed) - Informed

## Team Action Plans

### Backend Team (Priority 1)

**Backend Team Lead**: [Name] **Completion Deadline**: 24-48 hours

**Tasks:**

1. **Immediate (0-4 hours)**
   - Block access to /wp-json/wp/v2/users
   - Audit all REST API endpoints
   - Create list of all public APIs
2. **Short-term (1-2 days)**
   - Implement authentication for sensitive endpoints
   - Set up rate limiting
   - Add API request logging
3. **Medium-term (1 week)**
   - Conduct full API security review
   - Implement input validation
   - Set up automated security tests

**Resources:**

- 2 senior developers
- 1 junior for testing
- Security team consultations

**Completion Criteria:**

- ☐ All vulnerable endpoints blocked
- ☐ Authentication working correctly
- ☐ Logging configured
- ☐ Tests passing successfully

### Frontend Team (Priority 2)

**Frontend Team Lead**: [Name] **Completion Deadline**: 48-72 hours

**Tasks:**

1. **Immediate (0-8 hours)**
   - Implement basic security headers
   - Add CSP in report-only mode
   - Test on staging
2. **Short-term (1-3 days)**

   - Add SRI for all external resources
   - Switch CSP to enforcement mode
   - Update build process
3. **Medium-term (1 week)**
   - Automate SRI generation
   - Set up CSP violation monitoring
   - Train team on best practices

**Resources:**

- 2 frontend developers
- 1 DevOps for CI/CD setup
- QA team support

**Completion Criteria:**

- ☐ CSP configured and working
- ☐ SRI added for all resources
- ☐ No browser violations
- ☐ CI/CD checks headers

### DevOps Team (Priority 2)

**DevOps Team Lead**: [Name] **Completion Deadline**: 72 hours

**Tasks:**

1. **Immediate (0-4 hours)**
   - Apply security configurations to web servers
   - Hide server software versions
   - Configure forced HTTPS
2. **Short-term (1-3 days)**
   - Set up centralized logging
   - Implement security monitoring
   - Automate configuration checks
3. **Medium-term (1 week)**
   - Set up security event alerting
   - Implement Infrastructure as Code
   - Audit all servers

**Resources:**

- 2 DevOps engineers
- 1 SRE for monitoring
- Production server access

**Completion Criteria:**

- ☐ Security headers configured
- ☐ HTTPS enforced

- ☐ Monitoring operational
- ☐ Alerts configured

## Communication Plan

### Daily Standups (During Crisis)

**Time**: 9:00 AM daily **Participants**: All team leads + Security lead **Format**: 15 minutes maximum

**Structure:**

1. **Critical task status** (5 min)
2. **New issues and blockers** (5 min)
3. **Daily plans** (3 min)
4. **Questions and coordination** (2 min)

### Weekly Retrospectives

**Time**: Friday, 4:00 PM **Participants**: Extended team **Duration**: 1 hour

**Agenda:**

1. **What went well** (15 min)
2. **What can be improved** (20 min)
3. **Security lessons learned** (15 min)
4. **Next week plans** (10 min)

### Communication Channels

**Slack Channels:**

- #security-incident - Emergency notifications
- #security-general - General security questions
- #security-updates - Updates and patches

**Email Lists:**

- security-team@company.com - Security team
- team-leads@company.com - All team leads
- security-alerts@company.com - Critical alerts

# Conclusion and Next Steps

## Critical Actions for Next 72 Hours

**Day 1 (0-24 hours):**

- Backend: Block vulnerable API endpoints
- DevOps: Apply basic security headers
- All teams: Set up emergency communication

**Day 2 (24-48 hours):**

- Frontend: Implement CSP and basic headers
- Backend: Complete API security audit
- QA: Validate all changes

**Day 3 (48-72 hours):**

- DevOps: Set up monitoring and alerting
- Frontend: Add SRI for external resources
- All teams: Document changes

## Medium-term Goals (1-4 weeks)

**Week 1:**

- Complete implementation of all basic security measures
- Set up automated monitoring
- Train teams on security procedures

**Week 2-3:**

- Implement advanced monitoring tools
- Automate security checks in CI/CD
- Conduct internal penetration testing

**Week 4:**

- Complete audit of implemented measures
- Document all procedures
- Plan long-term strategy

## Success Metrics

**Technical Metrics:**

- Threat detection time: < 1 hour
- Incident response time: < 4 hours
- Security header coverage: 100%
- Vulnerability count: 90% reduction

**Process Metrics:**

- Security procedure compliance: > 95%
- Security task completion: within SLA
- Documentation quality: complete coverage
- Team knowledge level: regular testing

## Resources and Support

**Internal Resources:**

- Security team: consultations and support
- DevOps team: infrastructure support
- QA team: testing and validation

**External Resources:**

- Security consultants (if needed)
- Specialized monitoring tools
- Training materials and courses

**Emergency Contacts:**

- Security Lead: [contact details]
- DevOps Lead: [contact details]
- On-call Administrator: [contact details]

.

*This document is a living guide and should be updated as threats evolve and our security processes mature. All team leads are responsible for keeping it current in their areas of expertise.*