Security Configuration Guide

Comprehensive security setup and best practices for the Course Platform deployment.

Security Overview

This guide covers all security aspects of the deployment:
- Server hardening
- SSL/TLS configuration
- Firewall setup
- Application security
- Database security
- Monitoring and alerting

Server Security

Initial Server Hardening

1. **Update System**
   ```
   bash
   sudo apt update && sudo apt upgrade -y
   sudo apt autoremove -y
   ```

2. **Create Non-Root User**
   ```bash
   # Create deployment user
   sudo adduser deploy
   sudo usermod -aG sudo deploy
   sudo usermod -aG docker deploy

# Setup SSH key authentication
sudo mkdir -p /home/deploy/.ssh
sudo cp ~/.ssh/authorized_keys /home/deploy/.ssh/
sudo chown -R deploy:deploy /home/deploy/.ssh
sudo chmod 700 /home/deploy/.ssh
sudo chmod 600 /home/deploy/.ssh/authorized_keys
```

1. **Disable Root Login**
   ```bash
   sudo nano /etc/ssh/sshd_config

# Add/modify these lines:
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys
   ```

sudo systemctl restart sshd
```

1. **Configure SSH Security**
   ```bash
   # /etc/ssh/sshd_config
   Port 22
   Protocol 2
   HostKey /etc/ssh/ssh_host_rsa_key
   HostKey /etc/ssh/ssh_host_dsa_key
   HostKey /etc/ssh/ssh_host_ecdsa_key
   HostKey /etc/ssh/ssh_host_ed25519_key

# Authentication
LoginGraceTime 60
MaxAuthTries 3
MaxSessions 2

# Disable unused authentication methods
ChallengeResponseAuthentication no
KerberosAuthentication no
GSSAPIAuthentication no

sudo systemctl restart sshd
```

Firewall Configuration (UFW)

1. **Basic Firewall Setup**
   ```bash
   # Reset firewall rules
   sudo ufw –force reset

# Default policies
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Allow SSH (change port if needed)
sudo ufw allow 22/tcp

# Allow HTTP and HTTPS
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# Allow specific application ports (if needed)
sudo ufw allow 3000/tcp comment “Next.js Dev”
sudo ufw allow 5432/tcp comment “PostgreSQL”

# Enable firewall
sudo ufw –force enable
```

1. **Advanced Firewall Rules**
   ```bash

```
# Rate limiting for SSH
sudo ufw limit ssh

# Allow from specific IP ranges (adjust as needed)
sudo ufw allow from 10.0.0.0/8
sudo ufw allow from 172.16.0.0/12
sudo ufw allow from 192.168.0.0/16

# Block specific countries (example)
# sudo ufw deny from 1.2.3.0/24

# Check status
sudo ufw status verbose
```

Fail2Ban Configuration

1. **Install and Configure Fail2Ban**

```
bash
sudo apt install fail2ban -y

# Create local configuration
sudo cp /etc/fail2ban/jail.conf /etc/fail2ban/jail.local
```

1. **Custom Jail Configuration**

```
bash
  sudo nano /etc/fail2ban/jail.d/custom.conf
```

```ini
[DEFAULT]
bantime = 3600
findtime = 600
maxretry = 3

[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600

[nginx-http-auth]
enabled = true
filter = nginx-http-auth
port = http,https
logpath = /var/log/nginx/error.log
maxretry = 3
bantime = 3600

[nginx-limit-req]
enabled = true
filter = nginx-limit-req
port = http,https
```

logpath = /var/log/nginx/error.log
maxretry = 10
findtime = 600
bantime = 7200

[nginx-badbots]
enabled = true
filter = nginx-badbots
port = http,https
logpath = /var/log/nginx/access.log
maxretry = 2
bantime = 86400
```

1. **Custom Filters**
   `bash`
   `# Create custom filter for application`
   `sudo nano /etc/fail2ban/filter.d/course-platform.conf`

`ini`
`[Definition]`
`failregex = ^<HOST> - - \[.*\] "(GET|POST|HEAD).*" (401|403|404|500) .*$`
`ignoreregex =`

# SSL/TLS Security

## SSL Configuration

1. **Strong SSL Configuration**
   ```nginx
   # /etc/nginx/ssl.conf
   ssl_protocols TLSv1.2 TLSv1.3;
   ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384;
   ssl_prefer_server_ciphers off;
   ssl_session_cache shared:SSL:10m;
   ssl_session_timeout 10m;
   ssl_session_tickets off;

# OCSP Stapling
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;

# Security headers
add_header Strict-Transport-Security “max-age=31536000; includeSubDomains; preload” always;
add_header X-Frame-Options “SAMEORIGIN” always;
add_header X-Content-Type-Options “nosniff” always;
add_header X-XSS-Protection “1; mode=block” always;

add_header Referrer-Policy “strict-origin-when-cross-origin” always;
```

1. **Certificate Management**
   ```bash
   # Auto-renewal setup
   sudo crontab -e

# Add this line for automatic renewal
0 12 * * * /usr/bin/certbot renew –quiet

# Test renewal
sudo certbot renew –dry-run
```

1. **SSL Testing**
   ```bash
   # Test SSL configuration
   openssl s_client -connect example.com:443 -servername example.com

# Check certificate expiry
echo | openssl s_client -servername example.com -connect example.com:443 2>/dev/null | openssl x509 -noout -dates
```

Application Security

Environment Variables Security

1. **Secure Environment Files**
   ```bash
   # Set proper permissions
   chmod 600 .env.
   *chown deploy:deploy .env.*

# Never commit environment files
echo “.env*” >> .gitignore
```

1. **Environment Variable Validation**
   ```typescript
   // app/lib/env.ts
   import { z } from ‘zod’;

const envSchema = z.object({
DATABASE_URL: z.string().url(),
NEXTAUTH_SECRET: z.string().min(32),
NEXTAUTH_URL: z.string().url(),
YOOKASSA_SHOP_ID: z.string(),
YOOKASSA_SECRET_KEY: z.string(),
});

export const env = envSchema.parse(process.env);
```
```

API Security

1. **Rate Limiting**
   ```typescript
   // app/lib/rate-limit.ts
   import { NextRequest } from 'next/server';

const rateLimitMap = new Map();

export function rateLimit(request: NextRequest, limit = 10, window = 60000) {
const ip = request.ip || 'anonymous';
const now = Date.now();
const windowStart = now - window;

```
  const requests = rateLimitMap.get(ip) || [];
  const validRequests = requests.filter((time: number) => time > windowStart);

  if (validRequests.length >= limit) {
    return false;
  }

  validRequests.push(now);
  rateLimitMap.set(ip, validRequests);
  return true;
```

}
```

1. **Input Validation**
   ```typescript
   // app/lib/validation.ts
   import { z } from 'zod';

export const userSchema = z.object({
email: z.string().email(),
password: z.string().min(8).regex(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)/),
name: z.string().min(2).max(50),
});

export const courseSchema = z.object({
title: z.string().min(1).max(200),
description: z.string().max(1000),
price: z.number().positive(),
});
```

1. **CORS Configuration**
   ```typescript
   // app/middleware.ts
   import { NextResponse } from 'next/server';
   import type { NextRequest } from 'next/server';

export function middleware(request: NextRequest) {
const response = NextResponse.next();
   ```

```
  // CORS headers
  response.headers.set('Access-Control-Allow-Origin', 'https://www.example.com');
  response.headers.set('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OP-
TIONS');
  response.headers.set('Access-Control-Allow-Headers', 'Content-Type, Authorization');

  return response;
```

}
```

# Database Security

## PostgreSQL Security

1. **Database Configuration**
   ```sql
   – Create dedicated users for each environment
   CREATE USER course_platform_dev WITH PASSWORD ‘strong_dev_password’;
   CREATE USER course_platform_staging WITH PASSWORD ‘strong_staging_password’;
   CREATE USER course_platform_prod WITH PASSWORD ‘strong_prod_password’;

– Grant minimal required permissions
GRANT CONNECT ON DATABASE course_shop_platform_dev TO course_platform_dev;
GRANT USAGE ON SCHEMA public TO course_platform_dev;
GRANT CREATE ON SCHEMA public TO course_platform_dev;
```

1. **Connection Security**
   ```bash
   # PostgreSQL configuration (postgresql.conf)
   listen_addresses = ‘localhost’
   ssl = on
   ssl_cert_file = ‘/etc/ssl/certs/ssl-cert-snakeoil.pem’
   ssl_key_file = ‘/etc/ssl/private/ssl-cert-snakeoil.key’

# Authentication (pg_hba.conf)
local all all md5
host all all 127.0.0.1/32 md5
host all all ::1/128 md5
```

1. **Database Backup Security**
   ```bash
   # Encrypt backups
   pg_dump course_shop_platform_prod | gpg –cipher-algo AES256 –compress-algo 1 –symmetric –output
   backup.sql.gpg

# Secure backup storage
chmod 600 /var/www/course-platform/backups/
*chown deploy:deploy /var/www/course-platform/backups/*
```
```

Redis Security

1. **Redis Configuration**
   ```bash
   # /etc/redis/redis.conf
   bind 127.0.0.1
   requirepass your_strong_redis_password
   rename-command FLUSHDB ""
   rename-command FLUSHALL ""
   rename-command DEBUG ""
   rename-command CONFIG "CONFIG_b835c3f8a5d9e7f2"
   ```

Security Monitoring

Log Monitoring

1. **Centralized Logging**
   ```bash
   # Install log monitoring tools
   sudo apt install logwatch -y

# Configure logwatch
sudo nano /etc/logwatch/conf/logwatch.conf
```

1. **Security Log Analysis**
   ```bash
   # Create security monitoring script
   cat > /var/www/course-platform/scripts/security-monitor.sh << 'EOF'
   #!/bin/bash

LOG_FILE="/var/www/course-platform/logs/security.log"

# Check for failed login attempts
FAILED_LOGINS=$(grep "Failed password" /var/log/auth.log | wc -l)
if [ $FAILED_LOGINS -gt 10 ]; then
echo "$(date): High number of failed login attempts: $FAILED_LOGINS" >> $LOG_FILE
fi

# Check for suspicious nginx requests
SUSPICIOUS_REQUESTS=$(grep -E "(sql|union|select|script|alert)" /var/log/nginx/access.log | wc -l)
if [ $SUSPICIOUS_REQUESTS -gt 0 ]; then
echo "$(date): Suspicious requests detected: $SUSPICIOUS_REQUESTS" >> $LOG_FILE
fi

# Check disk space
DISK_USAGE=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')
if [ $DISK_USAGE -gt 90 ]; then
echo "$(date): High disk usage: ${DISK_USAGE}%" >> $LOG_FILE
fi
EOF

chmod +x /var/www/course-platform/scripts/security-monitor.sh
   ```

```
# Add to crontab
(crontab -l; echo "*/15 * * * * /var/www/course-platform/scripts/security-monitor.sh") | crontab -
```

Intrusion Detection

1. **File Integrity Monitoring**
   ```bash
   # Install AIDE
   sudo apt install aide -y

# Initialize database
sudo aideinit

# Create monitoring script
cat > /var/www/course-platform/scripts/file-integrity.sh << 'EOF'
#!/bin/bash

# Run AIDE check
sudo aide --check

# Check critical files
find /var/www/course-platform -name ".env" -newer /tmp/last-check 2>/dev/null
find /etc/nginx -newer /tmp/last-check 2>/dev/null
find /etc/ssh -newer /tmp/last-check 2>/dev/null

# Update timestamp
touch /tmp/last-check
EOF

chmod +x /var/www/course-platform/scripts/file-integrity.sh
```

Incident Response

Security Incident Checklist

1. **Immediate Response**
   - [ ] Identify the scope of the incident
   - [ ] Isolate affected systems
   - [ ] Preserve evidence
   - [ ] Notify stakeholders

2. **Investigation**
   - [ ] Analyze logs for attack vectors
   - [ ] Check for data breaches
   - [ ] Identify compromised accounts
   - [ ] Document findings

3. **Recovery**
   - [ ] Patch vulnerabilities
   - [ ] Reset compromised credentials
   - [ ] Restore from clean backups
   - [ ] Update security measures

4. **Post-Incident**
   - [ ] Conduct lessons learned session
   - [ ] Update security procedures
   - [ ] Implement additional monitoring
   - [ ] Train team on new procedures

Emergency Contacts

```
# Create emergency contact script
cat > /var/www/course-platform/scripts/emergency-notify.sh << 'EOF'
#!/bin/bash

INCIDENT_TYPE="$1"
SEVERITY="$2"
MESSAGE="$3"

# Email notification
echo "Security Incident: $INCIDENT_TYPE
Severity: $SEVERITY
Message: $MESSAGE
Time: $(date)
Server: $(hostname)" | mail -s "SECURITY ALERT: $INCIDENT_TYPE" admin@example.com

# Slack notification (if webhook configured)
if [ -n "$SLACK_WEBHOOK_URL" ]; then
    curl -X POST -H 'Content-type: application/json' \
        --data "{\"text\":\"  Security Alert: $INCIDENT_TYPE - $MESSAGE\"}" \
        "$SLACK_WEBHOOK_URL"
fi
EOF

chmod +x /var/www/course-platform/scripts/emergency-notify.sh
```

Security Maintenance

Regular Security Tasks

1. **Daily Tasks**
   ```bash
   # Check security logs
   tail -f /var/www/course-platform/logs/security.log

# Review fail2ban status
sudo fail2ban-client status

# Check for security updates
sudo apt list –upgradable | grep -i security
```

1. **Weekly Tasks**
   ```bash
   # Update system packages
   sudo apt update && sudo apt upgrade -y

# Review firewall logs
sudo ufw status verbose
   ```

```
# Check SSL certificate expiry
sudo certbot certificates
```

1. **Monthly Tasks**
   ```bash
   # Security audit
   sudo lynis audit system

# Review user accounts
sudo cat /etc/passwd | grep -v nologin

# Check for rootkits
sudo rkhunter –check
```

Security Testing

1. **Vulnerability Scanning**
   ```bash
   # Install security tools
   sudo apt install nmap nikto -y

# Network scan
nmap -sS -O localhost

# Web vulnerability scan
nikto -h https://www.example.com
```

1. **SSL Testing**
   ```bash
   # Test SSL configuration
   testssl.sh https://www.example.com

# Check certificate chain
openssl s_client -connect example.com:443 -showcerts
```

Security Checklist

Server Security

- [ ] Non-root user created and configured
- [ ] SSH hardened (key-based auth, no root login)
- [ ] Firewall configured and enabled
- [ ] Fail2Ban installed and configured
- [ ] System updates automated
- [ ] Log monitoring configured

Application Security

- [ ] Environment variables secured
- [ ] Input validation implemented
- [ ] Rate limiting configured

- [ ] CORS properly configured
- [ ] Security headers implemented
- [ ] Error handling doesn't leak information

Database Security

- [ ] Database users have minimal permissions
- [ ] Connections encrypted
- [ ] Backups encrypted
- [ ] Access restricted to localhost
- [ ] Strong passwords enforced

SSL/TLS Security

- [ ] Strong cipher suites configured
- [ ] HSTS headers enabled
- [ ] Certificate auto-renewal working
- [ ] OCSP stapling enabled
- [ ] SSL Labs grade A or higher

Monitoring & Response

- [ ] Security monitoring scripts running
- [ ] Log analysis automated
- [ ] Incident response plan documented
- [ ] Emergency contacts configured
- [ ] Regular security testing scheduled

Next Steps: After completing security setup, proceed to Troubleshooting Guide (./TROUBLESHOOTING.md)