الخطة التشغيلية للأمن السيبراني: دليل التنفيذ لقادة الفرق والمديرين

كبير مسؤولي أمن المعلومات

2026-01-02

الملخص التنفيذي للمديرين

تقدم هذه الوثيقة **دليلاً عملياً للتنفيذ** لتدابير الأمن السيبراني على مستوى الفرق والأقسام. بناءً على تدقيق الأمان المُجرى في 30 ديسمبر 2025، تم اكتشاف ثغرات حرجة تتطلب **إجراءات تشغيلية فورية** من قادة الفرق والإدارة الوسطى.

تنبيه حرج: تم اكتشاف ثغرة تعرض البيانات. يجب على فرقكم تنفيذ التدابير الوقائية خلال 72 ساعة. ⚠

الأولويات الإدارية الرئيسية

الأولوية	المهمة	الفريق المسؤول	الموعد النهائي
حرج	حجب نقاط API المعرضة للخطر	فرق Backend/DevOps	24 ساعة
عالي 🟠	تنفيذ رؤوس الأمان	فرق Frontend	48 ساعة
متوسط	نشر أنظمة المراقبة	فرق SRE/Ops	أسبوع واحد
منخفض ℹ	تدريب الفرق على الأمان	جميع الفرق	أسبوعان

مصفوفة مسؤولية الفرق

الإجراءات الفورية (0-72 ساعة):

- **فرق Backend**: إغلاق ثغرات REST API
- **فرق DevOps**: تكوين إعدادات الخادم الآمنة
- **فرق QA**: التحقق من إصلاحات الأمان

المهام قصيرة المدى (1-4 أسابيع):

- **فرق Frontend**: تنفيذ CSP و SRI
- **فرق البنية التحتية**: إعداد مراقبة الأمان
- **فرق الأمان**: وضع إجراءات الاستجابة للحوادث

الوضع التقني وأولويات الفرق

الثغرات الحرجة حسب الفريق

فرق Backend/API - أولوية حرجة

المشكلة: تعرض البيانات الشخصية عبر REST API

- **الثغرة**: /wp-json/wp/v2/users قابل للوصول بدون مصادقة
- **المخاطر**: جميع بيانات المستخدمين متاحة علنياً
- **نقاط CVSS**: 9.1 (حرج)

الإجراءات الفورية:

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

قائمة مراجعة قائد فريق Backend:

- ☐ حجب الوصول فوراً إلى /wp-json/wp/v2/users
- ☐ تدقيق جميع نقاط REST API
- ☐ تنفيذ المصادقة للنقاط الحساسة
- ☐ إعداد تسجيل طلبات API
- ☐ اختبار التغييرات على بيئة التطوير

فرق Frontend - أولوية عالية

المشكلة: غياب رؤوس الأمان الأساسية

- **الثغرات**: لا يوجد CSP، X-Frame-Options، SRI
- **المخاطر**: هجمات XSS، clickjacking، هجمات سلسلة التوريد

الحلول العملية:

1. **سياسة أمان المحتوى (CSP)**

```html
<!-- إضافة إلى <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. **تكامل الموارد الفرعية (SRI)**

```html
<!-- لجميع الموارد الخارجية -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.6.0/jquery.min.js"
      integrity="sha384-
vtXRMe3mGCbOeY7l30aIg8H9p3GdeSe4IFlP6G8JMa7o7lXvnz3GFKzPxzJdPfGK"
      crossorigin="anonymous"></script>
```

قائمة مراجعة قائد فريق Frontend:

- ☐ تنفيذ رؤوس CSP
- ☐ إضافة SRI لجميع الموارد الخارجية
- ☐ تكوين X-Frame-Options: DENY
- ☐ اختبار التوافق مع المتصفحات
- ☐ تحديث pipeline CI/CD للتحقق من الرؤوس

فرق DevOps/البنية التحتية - أولوية عالية

المشكلة: تكوين خادم الويب غير آمن

- **الثغرات**: غياب HSTS، كشف إصدارات البرمجيات
- **المخاطر**: هجمات man-in-the-middle، استطلاع المعلومات

تكوينات الخادم:

Apache (.htaccess):

```
<IfModule mod_headers.c>
   # رؤوس الأمان الأساسية
   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"

   # إخفاء معلومات الخادم
   Header always unset Server
   Header always unset X-Powered-By
</IfModule>

# تعطيل توقيع الخادم
ServerTokens Prod
ServerSignature Off
```

Nginx:

```
server {
   # رؤوس الأمان
   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;

   # إخفاء الإصدار
   server_tokens off;
   more_clear_headers Server;
}
```

قائمة مراجعة قائد فريق DevOps:

- ☐ تطبيق تكوينات الأمان
- ☐ فرض HTTPS
- ☐ إخفاء إصدارات برمجيات الخادم
- ☐ إعداد تسجيل الأمان
- ☐ اختبار التكوينات

الأدوات و التقنيات

الأدوات الأساسية للفرق

المراقبة و التنبيهات

1. مراقبة سجلات الأمان

```
# تكوين rsyslog لأحداث الأمان
echo "local0.*    /var/log/security.log" >> /etc/rsyslog.conf
systemctl restart rsyslog
```

2. مراقبة ملفات التكوين

```
#!/bin/bash
# security_monitor.sh - مراقبة تغييرات التكوين
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. التحقق التلقائي من الرؤوس

```
#!/usr/bin/env python3
# header_check.py - التحقق من رؤوس الأمان
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"فحص الرؤوس لـ {url}:")
      for header, expected in required_headers.items():
         if header in headers:
            print(f"   {header}: {headers[header]}")
         else:
            print(f"x {header}: ("مفقود

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

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

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

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

أدوات التطوير

1. خطافات الأمان قبل الالتزام

```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. توليد SRI التلقائي

```javascript
// sri-generator.js - توليد SRI تلقائي
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);
    });
}

// الاستخدام في عملية البناء
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(`خطأ في ${url}:`, error.message);
        }
    }
}

updateSRI();
```

تكامل CI/CD

تكوين GitLab CI:

```
# .gitlab-ci.yml - الأمان في 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 # التحقق من أن SRI محدثة
  only:
    - main
```

تكوين GitHub Actions:

```
# .github/workflows/security.yml
name: فحوصات الأمان
```

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

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

    steps:
    - uses: actions/checkout@v3

    - name: فحص رؤوس الأمان
      run: |
        python3 scripts/header_check.py https://staging.example.com

    - name: تدقيق التبعيات
      run: |
        npm audit --audit-level moderate

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

العمليات والإجراءات

إجراءات الاستجابة للحوادث

المستوى 1: الاكتشاف (فرق التطوير)

مؤشرات الحادث:

- نشاط غير عادي في السجلات
- تنبيهات من أنظمة المراقبة
- تقارير المستخدمين عن سلوك مشبوه
- فشل فحوصات الأمان التلقائية

الإجراءات الفورية (أول 15 دقيقة):

1. **الهدوء** - توثيق جميع الإجراءات
2. **العزل** - تعطيل الخدمات المشبوهة
3. **الإخطار** - إبلاغ قائد الفريق وفريق الأمان
4. **الحفظ** - أخذ لقطات من السجلات وحالة النظام

قالب الإخطار:

حادث أمني
الوقت: [YYYY-MM-DD HH:MM]
المكتشف: [الاسم]
النظام: [اسم النظام/الخدمة]
الوصف: [وصف موجز للمشكلة]
الإجراءات المتخذة: [ما تم فعله]
الحالة: [نشط/محتوى/محلول]

المستوى 2: التحليل (قادة الفرق)

إجراء التحليل (30-60 دقيقة):

1. **جمع المعلومات**

```
# سكريبت جمع بيانات الحادث
#!/bin/bash
INCIDENT_ID=$(date +%Y%m%d_%H%M%S)
INCIDENT_DIR="/var/log/incidents/$INCIDENT_ID"

mkdir -p $INCIDENT_DIR

# جمع السجلات
cp /var/log/apache2/access.log $INCIDENT_DIR/
cp /var/log/apache2/error.log $INCIDENT_DIR/
cp /var/log/security.log $INCIDENT_DIR/

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

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

echo "تم جمع بيانات الحادث في $INCIDENT_DIR"
```

2. **تصنيف الحادث**

- **المستوى 1**: إعلامي (تسجيل، مراقبة)
- **المستوى 2**: تحذير (تهديد محتمل)
- **المستوى 3**: حرج (هجوم نشط، تسريب بيانات)

3. **قرارات التصعيد**

المستوى 1 ← قائد الفريق ← التوثيق
المستوى 2 ← قائد الفريق + الأمان ← مراقبة معززة
المستوى 3 ← تصعيد فوري ← الإدارة + خبراء خارجيون

إجراءات تحديث الأمان

الفحوصات الأسبوعية (كل يوم اثنين)

قائمة مراجعة قائد الفريق:

المراجعة الأسبوعية للأمان

تحديثات النظام
- [ ] فحص التحديثات الأمنية المتاحة
- [ ] جدولة تثبيت التصحيحات الحرجة
- [ ] تحديث تبعيات المشروع (npm, composer, pip)

المراقبة
- [ ] مراجعة سجلات الأمان للأسبوع
- [ ] تحليل تنبيهات المراقبة
- [ ] التحقق من وظائف النسخ الاحتياطي

التكوينات
- [ ] التحقق من رؤوس الأمان
- [ ] فحص انتهاء شهادات SSL
- [ ] تدقيق صلاحيات وصول المستخدمين

الفريق
- [ ] مناقشة حوادث الأمان في الاستعراض
- [ ] تخطيط تدريب الأمان
- [ ] تحديث وثائق الإجراءات

التدقيقات الشهرية

إجراء مديري الأقسام:

1. **التدقيق التقني**

```
#!/bin/bash
# monthly_security_audit.sh
echo "=== "=== التدقيق الأمني الشهري
echo "التاريخ: $(date)"

# فحص المستخدمين بصلاحيات sudo
echo "المستخدمون بصلاحيات sudo:"
grep -Po '^sudo.+:\K.*$' /etc/group

# فحص المنافذ المفتوحة
echo "":المنافذ المفتوحة
nmap -sT -O localhost

# فحص محاولات تسجيل الدخول الفاشلة
echo "":محاولات تسجيل الدخول الفاشلة هذا الشهر
grep "Failed password" /var/log/auth.log | wc -l

# فحص حجم سجلات الأمان
echo "":حجم سجلات الأمان
du -sh /var/log/security.log
```

2. **تدقيق عمليات الفريق**

- الامتثال لإجراءات الأمان
- فعالية الاستجابة للحوادث
- جودة التوثيق
- مستوى معرفة الفريق

المراقبة والتحكم

أنظمة المراقبة للفرق

المراقبة الأساسية (مطلوبة لجميع الفرق)

1. مراقبة التوفر والأداء

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

# فحص التوفر
if curl -s --head $WEBSITE | head -n 1 | grep -q "200 OK"; then
   echo "$(date): $WEBSITE - متاح" >> $LOG_FILE
else
   echo "$(date): $WEBSITE - غير متاح" >> $LOG_FILE
   # إرسال تنبيه
   echo "الموقع غير متاح!" | mail -s "تنبيه: عدم توفر الموقع" admin@company.com
fi

# فحص رؤوس الأمان
HEADERS=$(curl -s -I $WEBSITE)
if echo "$HEADERS" | grep -q "X-Content-Type-Options"; then
   echo "$(date): رؤوس الأمان - متاحة" >> $LOG_FILE
else
   echo "$(date): رؤوس الأمان - مفقودة" >> $LOG_FILE
fi
```

2. مراقبة سجلات الأمان

```
#!/bin/bash
# log_monitor.sh - مراقبة النشاط المشبوه
SECURITY_LOG="/var/log/security.log"
ALERT_EMAIL="security@company.com"

# البحث عن أنماط مشبوهة
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 "تم اكتشاف نشاط مشبوه: $pattern" | \
        mail -s "تنبيه: نشاط مشبوه" $ALERT_EMAIL
    fi
done
```

المراقبة المتقدمة (لفرق DevOps)

1. تكوين Prometheus + Grafana

```
# 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. مقاييس الأمان المخصصة

```
#!/usr/bin/env python3
# security_metrics.py - مقاييس مخصصة لـ Prometheus
from prometheus_client import start_http_server, Counter, Gauge
import time
import requests
import re

# المقاييس
failed_logins = Counter('failed_logins_total', 'إجمالي محاولات تسجيل الدخول الفاشلة')
security_headers = Gauge('security_headers_present', 'وجود رؤوس الأمان', ['header'])
response_time = Gauge('security_check_response_time', 'وقت استجابة فحص الأمان')

def check_security_headers(url):
    """فحص رؤوس الأمان"""
    try:
        response = requests.get(url, timeout=10)
        headers = response.headers
```

```
        # فحص الرؤوس المهمة
        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"خطأ في الفحص: {e}")
        return 0

def count_failed_logins():
    """عد محاولات تسجيل الدخول الفاشلة"""
    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"خطأ في قراءة السجلات: {e}")

if __name__ == '__main__':
    # بدء خادم HTTP للمقاييس
    start_http_server(8000)

    while True:
        # تحديث المقاييس كل 60 ثانية
        response_time_val = check_security_headers('https://example.com')
        response_time.set(response_time_val)

        count_failed_logins()

        time.sleep(60)
```

لوحات المعلومات والتقارير

التقارير الأسبوعية لقادة الفرق

مولد التقارير التلقائي:

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

```python
def generate_weekly_report():
    """توليد التقرير الأسبوعي للأمان"""

    # فترة التقرير
    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': []
    }

    # تحليل سجلات الأمان
    security_events = analyze_security_logs(start_date, end_date)
    report['summary']['security_events'] = len(security_events)

    # فحص حالة الرؤوس
    headers_status = check_headers_compliance()
    report['metrics']['headers_compliance'] = headers_status

    # التوصيات
    if security_events:
        report['recommendations'].append("تم اكتشاف أحداث أمنية - يتطلب التحليل")

    if not headers_status['all_present']:
        report['recommendations'].append("ليست جميع رؤوس الأمان مكونة")

    return report

def analyze_security_logs(start_date, end_date):
    """تحليل سجلات الأمان للفترة"""
    events = []
    try:
        with open('/var/log/security.log', 'r') as f:
            for line in f:
                # تحليل بسيط - التنفيذ الحقيقي يحتاج تعقيد أكثر
                if 'SECURITY' in line:
                    events.append(line.strip())
    except FileNotFoundError:
        pass

    return events

def check_headers_compliance():
```

```
    """فحص امتثال رؤوس الأمان"""
    import requests

    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()

    # حفظ التقرير
    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"تم حفظ التقرير: {filename}")

    # إرسال عبر البريد الإلكتروني (اختياري)
    # send_report_email(report)
```

التدريب وتعليم الفريق

برامج التدريب للفرق

التدريب الأساسي على الأمان (إجباري لجميع الفرق)

1. وحدة الوعي الأمني

- **المدة:** 2 ساعة
- **التكرار:** شهرياً
- **المحتوى:**

- التعرف على التهديدات الشائعة
- أفضل ممارسات كلمات المرور
- التعرف على رسائل التصيد الاحتيالي
- إجراءات الإبلاغ عن الحوادث

2. ورشة الترميز الآمن

- **المدة:** 4 ساعات
- **التكرار:** ربع سنوي
- **المحتوى:**
  - ثغرات OWASP Top 10
  - تقنيات التحقق من المدخلات
  - إدارة الجلسات الآمنة
  - التشفير وإدارة المفاتيح

3. تمارين الاستجابة للحوادث

- **المدة:** 3 ساعات
- **التكرار:** نصف سنوي
- **المحتوى:**
  - محاكاة حوادث أمنية
  - تطبيق إجراءات الاستجابة
  - التواصل أثناء الأزمات
  - تحليل ما بعد الحادث

التدريب المتخصص حسب الفريق

فرق Backend:

برنامج تدريب أمان Backend

الأسبوع 1: أمان API
- [ ] مصادقة وتفويض API
- [ ] تقييد المعدل والحماية من DDoS
- [ ] تشفير البيانات المنقولة والمخزنة
- [ ] تسجيل وتدقيق API

الأسبوع 2: أمان قواعد البيانات
- [ ] منع حقن SQL
- [ ] إدارة الصلاحيات
- [ ] تشفير البيانات الحساسة
- [ ] النسخ الاحتياطي الآمن

الأسبوع 3: أمان الخادم
- [ ] تقوية نظام التشغيل
- [ ] إدارة التصحيحات
- [ ] مراقبة الأمان
- [ ] الاستجابة للحوادث

الأسبوع 4: التطبيق العملي
- [ ] مراجعة الكود الأمني
- [ ] اختبار الاختراق
- [ ] تحليل الثغرات
- [ ] وضع خطط التحسين

فرق Frontend:

برنامج تدريب أمان Frontend

الأسبوع 1: أمان المتصفح
- [ ] سياسة أمان المحتوى (CSP)
- [ ] تكامل الموارد الفرعية (SRI)
- [ ] رؤوس الأمان
- [ ] إدارة ملفات تعريف الارتباط

الأسبوع 2: منع الهجمات
- [ ] حماية من XSS
- [ ] منع CSRF
- [ ] حماية من Clickjacking
- [ ] التحقق من المدخلات

الأسبوع 3: أمان التطبيقات
- [ ] إدارة الحالة الآمنة
- [ ] التشفير من جانب العميل
- [ ] أمان التخزين المحلي
- [ ] التواصل الآمن مع API

الأسبوع 4: أدوات وأتمتة
- [ ] أدوات فحص الأمان
- [ ] تكامل CI/CD
- [ ] اختبار الأمان التلقائي
- [ ] مراقبة الأداء الأمني

فرق DevOps:

برنامج تدريب أمان DevOps

الأسبوع 1: أمان البنية التحتية
- [ ] تقوية الخوادم
- [ ] إدارة الشبكات الآمنة
- [ ] مراقبة الأمان
- [ ] إدارة الهوية والوصول

الأسبوع 2: أمان الحاويات
- [ ] أمان Docker
- [ ] أمان Kubernetes
- [ ] فحص صور الحاويات
- [ ] إدارة الأسرار

الأسبوع 3: أمان CI/CD
- [ ] تأمين pipelines
- [ ] فحص الكود التلقائي
- [ ] إدارة التبعيات
- [ ] نشر آمن

الأسبوع 4: المراقبة والاستجابة
- [ ] أنظمة SIEM
- [ ] تحليل السجلات
- [ ] الاستجابة للحوادث
- [ ] التعافي من الكوارث

تقييم المهارات والشهادات

نظام تقييم المهارات

1. اختبارات دورية

```python
#!/usr/bin/env python3
# skill_assessment.py - تقييم مهارات الأمان
import json
import datetime

class SecuritySkillAssessment:
    def __init__(self):
        self.questions = {
            'backend': [
                {
                    'question': 'ما هي أفضل طريقة لمنع حقن SQL؟',
                    'options': ['استخدام Prepared Statements', 'تشفير البيانات', 'جميع ما سبق', 'تقييد الوصول'],
                    'correct': 1,
                    'points': 10
                },
                {
                    'question': 'أي من التالي يُعتبر أفضل ممارسة لتخزين كلمات المرور؟',
                    'options': ['MD5', 'SHA1', 'bcrypt', 'Base64'],
                    'correct': 2,
                    'points': 15
                }
            ],
            'frontend': [
                {
                    'question': 'ما هو الغرض من Content Security Policy؟',
                    'options': ['منع XSS', 'تشفير البيانات', 'إدارة الذاكرة', 'تحسين الأداء'],
                    'correct': 1,
                    'points': 10
                },
                {
```

```
                'question': 'متى يجب استخدام SRI؟',
                'options': ['مع', 'مع الموارد الخارجية فقط', 'مع جميع الموارد', 'غير ضروري فقط'],
                'correct': 1,
                'points': 15
            }
        ]
    }

    def conduct_assessment(self, team_type, employee_id):
        """إجراء تقييم للموظف"""
        if team_type not in self.questions:
            return {'error': 'نوع فريق غير صحيح'}

        questions = self.questions[team_type]
        score = 0
        total_points = sum(q['points'] for q in questions)

        # محاكاة الإجابات (في التطبيق الحقيقي، ستأتي من واجهة المستخدم)
        answers = [1, 2]  # إجابات تجريبية

        for i, question in enumerate(questions):
            if i < len(answers) and answers[i] == question['correct']:
                score += question['points']

        percentage = (score / total_points) * 100

        result = {
            'employee_id': employee_id,
            'team_type': team_type,
            'score': score,
            'total_points': total_points,
            'percentage': percentage,
            'date': datetime.datetime.now().isoformat(),
            'status': 'pass' if percentage >= 70 else 'fail'
        }

        return result

    def generate_certificate(self, assessment_result):
        """توليد شهادة للموظف الناجح"""
        if assessment_result['status'] == 'pass':
            certificate = {
                'certificate_id': f"SEC-{assessment_result['employee_id']}-
{datetime.datetime.now().strftime('%Y%m%d')}",
                'employee_id': assessment_result['employee_id'],
                'skill_area': f"أمان {assessment_result['team_type']}",
                'score': assessment_result['percentage'],
                'issue_date': assessment_result['date'],
```

```
            'valid_until': (datetime.datetime.now() + datetime.timedelta(days=365)).isoformat()
        }
        return certificate
    return None

# استخدام النظام
if __name__ == '__main__':
    assessor = SecuritySkillAssessment()

    # تقييم موظف backend
    result = assessor.conduct_assessment('backend', 'EMP001')
    print(f"نتيجة التقييم: {result}")

    # توليد شهادة إذا نجح
    certificate = assessor.generate_certificate(result)
    if certificate:
        print(f"تم إصدار الشهادة: {certificate}")
```

2. مسارات التطوير المهني

```
## مسارات التطوير المهني في الأمان

### المستوى المبتدئ (0-6 أشهر)
- [ ] أساسيات الأمن السيبراني
- [ ] التعرف على التهديدات الشائعة
- [ ] أفضل ممارسات الترميز الآمن
- [ ] استخدام أدوات الأمان الأساسية

### المستوى المتوسط (6-18 شهر)
- [ ] تحليل الثغرات المتقدم
- [ ] تصميم الأنظمة الآمنة
- [ ] إدارة الحوادث الأمنية
- [ ] تدقيق الأمان

### المستوى المتقدم (18+ شهر)
- [ ] قيادة فرق الأمان
- [ ] تطوير استراتيجيات الأمان
- [ ] البحث في التهديدات الجديدة
- [ ] التدريب والإرشاد

### الشهادات المستهدفة
- CompTIA Security+
- CISSP (Certified Information Systems Security Professional)
- CEH (Certified Ethical Hacker)
- CISM (Certified Information Security Manager)
```

التعاون بين الفرق والتفويض

مصفوفة المسؤولية (RACI)

المهمة	Backend	Frontend	DevOps	QA	الأمان
حجب ثغرات API	R	I	C	A	C
تنفيذ CSP/SRI	I	R	C	A	C
تكوين رؤوس الخادم	I	I	R	A	C
مراقبة الأمان	I	I	R	I	A
تدريب الفريق	C	C	C	C	R
الاستجابة للحوادث	C	C	C	C	R

الرموز:

- **R** (Responsible) - مسؤول عن التنفيذ
- **A** (Accountable) - مساءل عن النتائج
- **C** (Consulted) - مستشار
- **I** (Informed) - مُطلع

خطط العمل للفرق

فريق Backend (الأولوية 1)

قائد فريق Backend: [الاسم] **موعد الإنجاز**: 24-48 ساعة

المهام:

1. **فوري (0-4 ساعات)**
   - حجب الوصول إلى /wp-json/wp/v2/users
   - تدقيق جميع نقاط REST API
   - إنشاء قائمة بجميع APIs العامة
2. **قصير المدى (1-2 يوم)**
   - تنفيذ المصادقة للنقاط الحساسة
   - إعداد تقييد المعدل
   - إضافة تسجيل طلبات API
3. **متوسط المدى (أسبوع واحد)**
   - إجراء مراجعة أمان شاملة لـ API
   - تنفيذ التحقق من المدخلات
   - إعداد اختبارات الأمان التلقائية

الموارد:

- 2 مطور أول
- 1 مطور مبتدئ للاختبار
- استشارات من فريق الأمان

معايير الإنجاز:

- ☐ جميع النقاط المعرضة للخطر محجوبة

- ☐ المصادقة تعمل بشكل صحيح
- ☐ التسجيل مكون
- ☐ الاختبارات تمر بنجاح

فريق Frontend ((الأولوية 2

قائد فريق Frontend: [الاسم] **موعد الإنجاز**: 72-48 ساعة

المهام:

1. **فوري (0-8 ساعات)**
   - تنفيذ رؤوس الأمان الأساسية
   - إضافة CSP في وضع التقرير فقط
   - اختبار على بيئة التطوير
2. **قصير المدى (1-3 أيام)**
   - إضافة SRI لجميع الموارد الخارجية
   - تحويل CSP إلى وضع الإنفاذ
   - تحديث عملية البناء
3. **متوسط المدى (أسبوع واحد)**
   - أتمتة توليد SRI
   - إعداد مراقبة انتهاكات CSP
   - تدريب الفريق على أفضل الممارسات

الموارد:

- 2 مطور frontend
- 1 DevOps لإعداد CI/CD
- دعم فريق QA

معايير الإنجاز:

- ☐ CSP مكون ويعمل
- ☐ SRI مضاف لجميع الموارد
- ☐ لا توجد انتهاكات في المتصفحات
- ☐ CI/CD يفحص الرؤوس

فريق DevOps ((الأولوية 2

قائد فريق DevOps: [الاسم] **موعد الإنجاز**: 72 ساعة

المهام:

1. **فوري (0-4 ساعات)**
   - تطبيق تكوينات الأمان على خوادم الويب
   - إخفاء إصدارات برمجيات الخادم
   - تكوين الإجباري HTTPS
2. **قصير المدى (1-3 أيام)**
   - إعداد التسجيل المركزي

- تنفيذ مراقبة الأمان
- أتمتة فحوصات التكوين

3. **متوسط المدى (أسبوع واحد)**
   - إعداد تنبيهات أحداث الأمان
   - تنفيذ Infrastructure as Code
   - تدقيق جميع الخوادم

الموارد:

- 2 مهندس DevOps
- 1 SRE للمراقبة
- وصول لخوادم الإنتاج

معايير الإنجاز:

- ☐ رؤوس الأمان مكونة
- ☐ مفروض HTTPS
- ☐ المراقبة تعمل
- ☐ التنبيهات مكونة

خطة التواصل

الاجتماعات اليومية (أثناء الأزمة)

الوقت: 9:00 صباحاً يومياً **المشاركون**: جميع قادة الفرق + قائد الأمان **الشكل**: 15 دقيقة كحد أقصى

الهيكل:

1. (5 دقائق) **حالة المهام الحرجة**
2. (5 دقائق) **المشاكل والعوائق الجديدة**
3. (3 دقائق) **خطط اليوم**
4. (2 دقيقة) **الأسئلة والتنسيق**

الاستعراضات الأسبوعية

الوقت: الجمعة، 4:00 مساءً **المشاركون**: الفريق الموسع **المدة**: ساعة واحدة

جدول الأعمال:

1. (15 دقيقة) **ما سار بشكل جيد**
2. (20 دقيقة) **ما يمكن تحسينه**
3. (15 دقيقة) **دروس الأمان المستفادة**
4. (10 دقائق) **خطط الأسبوع القادم**

قنوات التواصل

قنوات Slack:

- #security-incident - الإخطارات الطارئة
- #security-general - أسئلة الأمان العامة
- #security-updates - التحديثات والتصحيحات

قوائم البريد الإلكتروني:

- فريق الأمان - security-team@company.com
- جميع قادة الفرق - team-leads@company.com
- التنبيهات الحرجة - security-alerts@company.com

الخلاصة والخطوات التالية

الإجراءات الحرجة للـ 72 ساعة القادمة

اليوم 1 (0-24 ساعة):

- Backend: حجب نقاط API المعرضة للخطر
- DevOps: تطبيق رؤوس الأمان الأساسية
- جميع الفرق: إعداد التواصل الطارئ

اليوم 2 (24-48 ساعة):

- Frontend: تنفيذ CSP والرؤوس الأساسية
- Backend: إكمال تدقيق أمان API
- QA: التحقق من جميع التغييرات

اليوم 3 (48-72 ساعة):

- DevOps: إعداد المراقبة والتنبيهات
- Frontend: إضافة SRI للموارد الخارجية
- جميع الفرق: توثيق التغييرات

الأهداف متوسطة المدى (1-4 أسابيع)

الأسبوع 1:

- تنفيذ كامل لجميع تدابير الأمان الأساسية
- إعداد المراقبة التلقائية
- تدريب الفرق على إجراءات الأمان

الأسبوع 2-3:

- تنفيذ أدوات المراقبة المتقدمة
- أتمتة فحوصات الأمان في CI/CD
- إجراء اختبار اختراق داخلي

الأسبوع 4:

- تدقيق شامل للتدابير المنفذة
- توثيق جميع الإجراءات
- تخطيط الاستراتيجية طويلة المدى

مقاييس النجاح

المقاييس التقنية:

- وقت اكتشاف التهديدات: < ساعة واحدة

- وقت الاستجابة للحوادث: < 4 ساعات
- تغطية رؤوس الأموال: 100%
- عدد الثغرات: انخفاض 90%

مقاييس العمليات:

- الامتثال لإجراءات الأمان: > 95%
- وقت إنجاز مهام الأمان: ضمن SLA
- جودة التوثيق: تغطية كاملة
- مستوى معرفة الفريق: اختبار منتظم

الموارد والدعم

الموارد الداخلية:

- فريق الأمان: الاستشارات والدعم
- فريق DevOps: دعم البنية التحتية
- فريق QA: الاختبار والتحقق

الموارد الخارجية:

- استشاريو الأمان (عند الحاجة)
- أدوات المراقبة المتخصصة
- مواد ودورات التدريب

جهات الاتصال للحالات الطارئة:

- قائد الأمان: [تفاصيل الاتصال]
- قائد DevOps: [[تفاصيل الاتصال]
- المدير المناوب: [تفاصيل الاتصال]

.

هذه الوثيقة دليل حي يجب تحديثه مع تطور التهديدات ونضج عملياتنا الأمنية. جميع قادة الفرق مسؤولون عن إبقائها محدثة في مجالات خبرتهم.