Hướng dẫn Security Workflow - Incremental Approach

Phương pháp: Kiểm tra và sửa dần theo tiến trình phát triển từng module

---

📊 Tổng quan Phương pháp

Tại sao chọn Incremental?

Tiêu chí	Global Hardening	Incremental (Khuyến nghị)
Thời gian ban đầu	8-15 giờ	Chia nhỏ theo sprint
Rủi ro build lỗi	Cao	Thấp (dễ kiểm soát)
Khả năng bỏ sót	Thấp	Thấp (nếu tracking tốt)
Phù hợp môi trường	Production ổn định	**Mạng nội bộ quân đội**
Tiến độ nhìn thấy	Khó theo dõi	Dễ dàng (dashboard)

🛠️ Công cụ Đã Cung cấp

1. Security Scanner

```
yarn tsx scripts/diagnostics/security-scan.ts
```

Chức năng:
- Quét tất cả 176 API routes
- Kiểm tra 4 tiêu chuẩn:
- ✅ `handleError` / `handleApiError`
- ✅ `logger` (structured logging)
- ✅ `api-guards` (requireAuth, requireRole)
- ✅ `validators` (Zod schemas)
- Phân loại theo module (Auth, Submissions, Reviews, ...)
- Tính toán security score (0-100)
- Tạo priority list (CRITICAL, HIGH, MEDIUM, LOW)

Output:
- Console report
- `logs/security-scan-[timestamp].json` (chi tiết)
- `logs/security-scan-latest.txt` (tóm tắt)

---

2. Security Dashboard

```
yarn tsx scripts/diagnostics/security-dashboard.ts
```

Chức năng:
- Tạo HTML dashboard trực quan
- Hiển thị:
- Overall statistics (FULL, PARTIAL, BASIC, NONE)
- Progress bar
- Module breakdown (16 modules)
- Priority action list

Output:
- `logs/security-dashboard.html`

Cách xem:

```
# Open in browser
xdg-open logs/security-dashboard.html

# Or
firefox logs/security-dashboard.html
```

---

3. Prebuild Check

```
yarn tsx scripts/diagnostics/prebuild-check.ts
```

Chức năng:
- Chạy trước khi build
- Kiểm tra thếreshold:
- CRITICAL routes < 10 (hiện tại: 34 🔴)
- Overall coverage > 50% (hiện tại: 0% 🔴)
- **Hiện tại:** Chỉ warning, không block build
- **Tương lai:** Sẽ block khi gần production

---

4. Module Checklist

```
scripts/diagnostics/module-checklist.md
```

Thông tin:
- Template tracking cho mỗi module
- Priority list
- Workflow instructions

📋 Quy trình Thực hiện

Bước 1: Chọn Module Ưu tiên

Priority Order:

```
🔴 CRITICAL (34 routes)
  1. Auth (16 routes) - Entry points, token management
  2. Admin (14 routes) - User management, role escalation
  3. Submissions (8 routes) - File uploads, data creation

🟠 HIGH (13 routes)
  4. Reviews (3 routes) - Sensitive reviewer data
  5. Editor (1 route) - Workflow management
  6. Users (9 routes) - Profile, password changes

🟡 MEDIUM (128 routes)
  7. Articles, Issues, Statistics, etc.

🟢 LOW
  8. Public routes, search
```

Bước 2: Chạy Security Scan

```
# Lần đầu tiên
cd /home/ubuntu/tapchi-hcqs/nextjs_space
yarn tsx scripts/diagnostics/security-scan.ts

# Xem dashboard
yarn tsx scripts/diagnostics/security-dashboard.ts
xdg-open logs/security-dashboard.html
```

Kết quả hiện tại:

```
FULL: 0 routes (0%)
PARTIAL: 1 routes (1%)
BASIC: 6 routes (3%)
NONE: 169 routes (96%)

Module scores:
  Auth: 15/100 (🔴 NEEDS WORK)
  Submissions: 21/100 (🔴 NEEDS WORK)
  Reviews: 10/100 (🔴 NEEDS WORK)
  ...
```

Bước 3: Hardening từng Route

Ví dụ: Hardening `/api/auth/refresh`

Trước khi hardening:

```
export async function POST(req: NextRequest) {
  try {
    const token = cookies().get('refresh-token');
    const user = await prisma.user.findUnique({ ... });
    return NextResponse.json({ success: true, user });
  } catch (error: any) {
    return NextResponse.json(
      { error: error.message },
      { status: 500 }
    );
  }
}
```

Sau khi hardening:

```
import { handleApiError } from '@/lib/error-handler';
import { logger } from '@/lib/logger';
import { requireAuth } from '@/lib/api-guards';

/**
 * POST /api/auth/refresh
 * Mô tả: Làm mới access token
 * Auth: Required (refresh token)
 */
export async function POST(req: NextRequest) {
  try {
    // Log request
    logger.info('Token refresh attempt', {
      context: 'API_AUTH_REFRESH'
    });

    const token = cookies().get('refresh-token');

    if (!token) {
      throw new ValidationError('Refresh token không tồn tại');
    }

    const user = await prisma.user.findUnique({ ... });

    if (!user) {
      throw new NotFoundError('Người dùng không tồn tại');
    }

    // Log success
    logger.info('Token refreshed successfully', {
      context: 'API_AUTH_REFRESH',
      userId: user.id
    });

    return successResponse({ user });

  } catch (error) {
    // Log error
    logger.error('Token refresh failed', {
      context: 'API_AUTH_REFRESH',
      error: error instanceof Error ? error.message : String(error)
    });

    return handleApiError(error, 'API_AUTH_REFRESH');
  }
}
```

Checklist:

- ✅ Import `handleApiError`, `logger`
- ✅ Thêm JSDoc comment
- ✅ Log request (`logger.info`)
- ✅ Use custom error classes (ValidationError, NotFoundError)
- ✅ Log success
- ✅ Log error trong catch block
- ✅ Return `handleApiError(error, context)`
- ✅ Use `successResponse()` helper

Bước 4: Test và Verify

```
# Build test
yarn build

# Chạy lại security scan
yarn tsx scripts/diagnostics/security-scan.ts

# Kiểm tra tiến độ
yarn tsx scripts/diagnostics/security-dashboard.ts
```

Kết quả mong đợi:

```
Auth module:
  Before: 15/100 (0% coverage)
  After:  45/100 (20% coverage) ↑

  /api/auth/refresh: NONE → FULL ✅
```

Bước 5: Cập nhật Checklist

File: `scripts/diagnostics/module-checklist.md`

```
## Module: Authentication

| API Route | Methods | handleError | logger | guards | validator | Status | Ghi chú |
|-----------|---------|-------------|--------|--------|-----------|--------|----------
|
| `/api/auth/login` | POST | ✅ | ✅ | ✅ | ✅ | FULL | Done |
| `/api/auth/register` | POST | ✅ | ✅ | ✅ | ✅ | FULL | Done |
| `/api/auth/refresh` | POST | ✅ | ✅ | ✅ | ❌ | **PARTIAL** | **Cần validator** |

**Tổng kết Module:**
- Tổng số routes: 16
- FULL: 2 (12.5%)
- PARTIAL: 1 (6.25%)
- BASIC: 13 (81.25%)
- Coverage: 12.5% ↑
```

Bước 6: Commit và Deploy

```
git add .
git commit -m "feat(security): hardening auth/refresh route

- Added handleApiError integration
- Added structured logging
- Custom error classes
- Updated security score: 15 -> 45

Module: Auth (12.5% coverage)"

git push
```

📋 Kế hoạch Thực hiện

Sprint 1: Auth Module (16 routes)

Thời gian: 2 sessions (~12 giờ)
Mục tiêu: 100% Auth coverage

Routes:

```
✅ /api/auth/login (Done)
✅ /api/auth/register (Done)
⚠️ /api/auth/refresh (Partial)
❌ /api/auth/logout
❌ /api/auth/me
❌ /api/auth/forgot-password
❌ /api/auth/reset-password
❌ /api/auth/verify-email
❌ /api/auth/2fa/*
❌ /api/auth/orcid/*
```

Output:
- Auth module: 0% → 100%
- Overall: 1% → 9%

Sprint 2: Submissions Module (8 routes)

Thời gian: 1 session (~6 giờ)
Mục tiêu: 100% Submissions coverage

Routes:

```
❌ /api/submissions (GET, POST)
❌ /api/submissions/[id] (GET, PUT, DELETE)
❌ /api/submissions/[id]/status
❌ /api/submissions/[id]/assign-reviewers
❌ /api/submissions/[id]/decision
❌ /api/author/submissions
```

Output:
- Submissions module: 0% → 100%
- Overall: 9% → 14%

Sprint 3: Reviews Module (3 routes)

Thời gian: 1 session (~3 giờ)

Output:
- Reviews module: 0% → 100%
- Overall: 14% → 16%

Sprint 4-12: Remaining modules

Thời gian: 8 sessions (~40 giờ)
Mục tiêu: 100% overall coverage

Timeline tổng:
- 12 sessions
- 2-3 tuần
- 100% coverage

📊 Theo dõi Tiến độ

Quick Commands

```
# 1. Chạy scan và xem dashboard
yarn tsx scripts/diagnostics/security-scan.ts && \
yarn tsx scripts/diagnostics/security-dashboard.ts && \
xdg-open logs/security-dashboard.html

# 2. Kiểm tra module cụ thể
cat logs/security-scan-latest.txt | grep "Auth"

# 3. Liệt kê priority routes
cat logs/security-scan-latest.txt | grep "CRITICAL"
```

Dashboard Metrics

Target by Sprint:

Sprint	Module	Target Coverage	Overall Coverage
0 (Current)	-	-	1.1%
1	Auth	100%	9%
2	Submissions	100%	14%
3	Reviews	100%	16%
4	Admin	100%	24%
5	Users	100%	29%
6-12	Remaining	100%	100%

⚠️ Lưu ý Quan trọng

1. Không Hardening Tất cả Cùng Lúc

- ❌ Đừng chạy automation cho 176 routes
- ❌ Đừng mass edit nhiều files
- ✅ Làm từng module, từng route
- ✅ Test sau mỗi thay đổi

2. Tracking Tiến độ

- ✅ Chạy security scan sau mỗi session
- ✅ Cập nhật module checklist
- ✅ Commit với message rõ ràng
- ✅ Xem dashboard thường xuyên

3. Quality > Speed

- ✅ Manual review tốt hơn automation sai
- ✅ Test kỹ trước khi commit
- ✅ Giữ code rõ ràng, dễ đọc

4. Documentation

- ✅ Cập nhật checklist liên tục
- ✅ Ghi chú vấn đề gặp phải
- ✅ Share knowledge với team

📚 Tài liệu Tham khảo

Security Frameworks

- `lib/error-handler.ts` - Error classes & handleError

- `lib/logger.ts` - Structured logging
- `lib/api-guards.ts` - Auth middleware
- `lib/validators.ts` - Zod schemas
- `lib/responses.ts` - Response helpers

Reports

- `SECURITY_AUDIT_REPORT.md` - Initial audit
- `FINAL_HARDENING_REPORT.md` - Automation attempt analysis
- `module-checklist.md` - Module tracking

Tools

- `scripts/diagnostics/security-scan.ts`
- `scripts/diagnostics/security-dashboard.ts`
- `scripts/diagnostics/prebuild-check.ts`

❓ FAQ

Q: Tại sao không dùng automation?

A: Automation failed do:
- Code structure phức tạp, nhiều patterns khác nhau
- Regex-based replacement không an toàn
- Cần AST parsing (phức tạp hơn)
- Manual approach an toàn hơn cho 176 routes

Q: Liệu có bỏ sót routes không?

A: KHÔNG, nếu:
- ✅ Chạy security-scan thường xuyên
- ✅ Xem dashboard mỗi tuần
- ✅ Cập nhật checklist
- ✅ Review priority list

Q: Mất bao lâu để hardening hết?

A:
- **Full coverage:** 12 sessions (~2-3 tuần)
- **Critical only (40 routes):** 5 sessions (~1 tuần)
- **Current:** 2 routes done, 174 remaining

Q: Có thể deploy trước khi hardening hết?

A: CÓ, nhưng:
- ⚠️ Nên hoàn thành Auth module trước (CRITICAL)
- ⚠️ Monitor logs chặt chẽ
- ⚠️ Có kế hoạch tăng cường security

Q: Tools này có chạy tự động không?

A: Hiện tại: KHÔNG (manual)

Tương lai có thể:

```
# Add to package.json scripts
"prebuild": "tsx scripts/diagnostics/prebuild-check.ts"

# Or CI/CD pipeline
- name: Security Check
  run: yarn tsx scripts/diagnostics/security-scan.ts
```

🎯 Kết luận

Phương pháp Incremental là tối ưu cho:
- ✅ Mạng nội bộ quân đội
- ✅ Hệ thống đang phát triển liên tục
- ✅ Cần theo dõi tiến độ rõ ràng
- ✅ Team có thể chia nhỏ công việc

Công cụ đã sẵn sàng:
- ✅ Security Scanner
- ✅ Visual Dashboard
- ✅ Prebuild Check
- ✅ Module Checklist
- ✅ Complete Documentation

Bắt đầu ngay:

```
cd /home/ubuntu/tapchi-hcqs/nextjs_space
yarn tsx scripts/diagnostics/security-scan.ts
yarn tsx scripts/diagnostics/security-dashboard.ts
xdg-open logs/security-dashboard.html
```

Chúc bạn hardening thành công! 🚀🔒

Người tạo: DeepAgent
Ngày: 28/12/2025
Version: 1.0
Liên hệ: See project documentation